1 Hello, Reader

I love technology. I've loved it since I was a little girl and my parents bought me an Erector Set that I used to build a giant (to me) robot out of small pierced pieces of metal. The robot was supposed to be powered by a miniature, battery-driven motor. I was an imaginative kid; I convinced myself that once this robot was built, it would move around the house as easily as I did, and I would have a new robot best friend. I would teach the robot to dance. It would follow me around the house, and (unlike my dog) it would play fetch.

I spent hours sitting on the red wool rug in the upstairs hallway of my parents' house, daydreaming and assembling the robot. I tightened dozens of nuts and bolts using the set's tiny, child-sized wrenches. The most exciting moment came when I was ready to plug in the motor. My mom and I made a special trip to the store to get the right batteries for the motor. We got home, and I raced upstairs to connect the bare wires to the gears and turn on my robot. I felt like Orville and Wilbur Wright at Kitty Hawk, launching a new machine and hoping it would change the world.

Nothing happened.

I checked the diagrams. I flicked the on/off switch a few times. I flipped the batteries. Still nothing. My robot didn't work. I went to get my mom.

"You need to come upstairs. My robot isn't working," I said sadly.

"Did you try turning it off and turning it on again?" my mom asked.

"I did that," I said.

"Did you try flipping the batteries?" she asked.

"Yes," I said. I was getting frustrated.

"I'll come look at it," she said. I grabbed her hand and pulled her upstairs. She tinkered with the robot for a little bit, looking at the directions and fiddling with the wiring and turning the switch on and off a few times. "It's not working," she said finally.

"Why not?" I asked. She could have just told me that the motor was broken, but my mother believed in complete explanations. She told me that the motor was broken, and then she also explained global supply chains and assembly lines and reminded me that I knew how factories worked because I liked to watch the videos on *Sesame Street* featuring huge industrial machines making packages of crayons.

"Things can go wrong when you make things," she explained. "Something went wrong when they made this motor, and it ended up in your kit anyway, and now we're going to get one that works." We called the Erector hotline number printed on the instructions, and the nice people at the toy company sent us a new motor in the mail. It arrived in a week or so, and I plugged it in, and my robot worked. By that point, it was anticlimactic. The robot worked, but not well. It could move slowly across the hardwood floor. It got stuck on the rug. It wasn't going to be my new best friend. After a few days, I took the robot apart to make the next project in the kit, a Ferris wheel.

I learned a few things from making this robot. I learned how to use tools to build technology, and that building things could be fun. I discovered that my imagination was powerful, but that the reality of technology couldn't measure up to what I imagined. I also learned that parts break.

A few years later, when I began writing computer programs, I discovered that these lessons from robot building translated well to the world of computer code. I could imagine vastly complex computer programs, but what the computer could *actually* do was often a letdown. I ran into many situations where programs didn't work because a part failed somewhere in the computer's innards. Nevertheless, I persisted, and I still love building and using technology. I have a vast number of social media accounts. I once hacked a crockpot to build a device for tempering twenty-five pounds of chocolate as part of a cooking project. I even built a computerized system to automatically water my garden.

Recently, however, I've become skeptical of claims that technology will save the world. For my entire adult life, I've been hearing promises about what technology can do to change the world for the better. I began studying computer science at Harvard in September 1991, months after Tim Berners-Lee launched the world's first website at CERN, the particle physics lab run by the European Organization for Nuclear Research. In my sophomore year, my roommate bought a NeXT cube, the same square black computer that Berners-Lee used as a web server at CERN. It was fun. My roommate got a high-speed connection in our dormitory suite, and we used his $5,000 computer to check our email. Another roommate, who had recently come out and was too young for Boston's gay bar scene, used the computer to hang out on online bulletin boards and meet boys. It was easy to believe that in the future, we would do everything online.

For youthful idealists of my generation, it was also easy to believe that the world we were creating online would be better and more just than the world we already had. In the 1960s, our parents thought they could make a better world by dropping out or living in communes. We saw that our parents had gone straight, and communes clearly weren't the answer—but there was this entire new, uncharted world of "cyberspace" that was ours for the making. The connection wasn't just metaphorical. The emerging Internet culture of the time was heavily influenced by the New Communalism movement of the 1960s, as Fred Turner writes in *From Counterculture to Cyberculture*, a history of digital utopianism.[1] Stewart Brand, the founder of the *Whole Earth Catalog*, laid out the connections between the counterculture and the personal computer revolution in an essay called "We Owe It All to the Hippies," in a 1995 special issue of *Time* magazine called "Welcome to Cyberspace."[2] The early Internet was deeply groovy.

By my junior year, I could make a web page or spin up a web server or write code in six different programming languages. For an undergraduate majoring in math, computer science, or engineering at the time, this was completely normal. For a woman, it wasn't. I was one of six undergraduate women majoring in computer science at a university of twenty thousand graduate and undergraduate students. I only knew two of the other women in computer science. The other three felt like a rumor. I felt isolated in all of the textbook ways that cause women to drop out of science, technology, engineering, and mathematics (STEM) careers. I could see what was broken inside the system, for me and for other women, but I didn't have the power to fix it. I switched my major.

I took a job as a computer scientist after college. My job was to make a simulator that was like a million bees with machine guns attacking all at once so that we could deploy the bees against a piece of software, to test that the software wouldn't go down when it was deployed. It was a good job, but I wasn't happy. Again, it felt like there was nobody around who looked like me or talked like me or was interested in the things that interested me. I quit to become a journalist.

Fast-forward a few years: I went back to computer science as a data journalist. *Data journalism* is the practice of finding stories in numbers and using numbers to tell stories. As a data journalist, I write code in order to commit acts of investigative journalism. I'm also a professor. It suits me. The gender balance is better, too.

Journalists are taught to be skeptical. We tell each other, "If your mother says she loves you, check it out." Over the years, I heard people repeat the same promises about the bright technological future, but I saw the digital world replicate the inequalities of the "real" world. For example, the percentage of women and minorities in the tech workforce never increased significantly. The Internet became the new public sphere, but friends and colleagues reported being harassed online more than they ever were before. My women friends who used online dating sites and apps received rape threats and obscene photos. Trolls and bots made Twitter a cacophony.

I started to question the promises of tech culture. I started to notice that the way people talk about technology is out of sync with what digital technology actually can do. Ultimately, everything we do with computers comes down to math, and there are fundamental limits to what we can (and should) do with it. I think that we have reached that limit. Americans have hit a point at which we are so enthusiastic about using technology for everything—hiring, driving, paying bills, choosing dates—that we have stopped demanding that our new technology is *good*.

Our collective enthusiasm for applying computer technology to every aspect of life has resulted in a tremendous amount of poorly designed technology. That badly designed technology is getting in the way of everyday life rather than making life easier. Simple things like finding a new friend's phone number or up-to-date email address have become time-consuming. The problem here, as in so many cases, is too much technology and not enough people. We turned over record-keeping to computational systems but fired all the humans who kept the information up-to-date. Now, since nobody goes through and makes sure all the contact information is accurate in every institutional directory, it is more difficult than ever to get in touch with people. As a journalist, a lot of my job involves reaching out to people I don't know. It's harder than it used to be, and it's more expensive, to contact anyone.

There's a saying: When all you have is a hammer, everything looks like a nail. Computers are our hammers. It's time to stop rushing blindly into the digital future and start making better, more thoughtful decisions about when and why to use technology.

Hence, this book.

This book is a guide for understanding the outer limits of what technology can do. It's about understanding the bleeding edge, where human achievement intersects with human nature. That edge is more like a cliff; beyond it lies danger.

The world is full of marvelous technology: Internet search, devices that recognize spoken commands, computers that can compete against human experts in games like Jeopardy! or Go. In celebrating these achievements, it's important not to get too carried away and assume that because we have cool technology, we can use it to solve every problem. In my university classes, one of the fundamental things I teach is that there are limits. Just as there are fundamental limits to what we know in mathematics and in science, so are there fundamental limits to what we can do with technology. There are also limits to what we *should* do with technology. When we look at the world only through the lens of computation, or we try to solve big social problems using technology alone, we tend to make a set of the same predictable mistakes that impede progress and reinforce inequality. This book is about how to understand the outer limits of what technology can do. Understanding these limits will help us make better choices and have collective conversations as a society about what we can do with tech and what we ought to do to make the world truly better for everyone.

I come to this conversation about social justice as a journalist. I specialize in a particular kind of data journalism, called *computational journalism* or *algorithmic accountability reporting*. An *algorithm* is a computational procedure for deriving a result, much like a recipe is a procedure for making a particular dish. Sometimes, algorithmic accountability reporting means writing code to investigate the algorithms that are being used increasingly to make decisions on our behalf. Other times, it means looking at badly designed technology or falsely interpreted data and raising a red flag.

One of the red flags I want to raise in this book is a flawed assumption that I call *technochauvinism*. Technochauvinism is the belief that tech is always the solution. Although digital technology has been an ordinary part of scientific and bureaucratic life since the 1950s, and everyday life since the 1980s, sophisticated marketing campaigns still have most people convinced that tech is something new and potentially revolutionary. (The tech revolution has already happened; tech is now mundane.)

Technochauvinism is often accompanied by fellow-traveler beliefs such as Ayn Randian meritocracy; technolibertarian political values; celebrating free speech to the extent of denying that online harassment is a problem; the notion that computers are more "objective" or "unbiased" because they distill questions and answers down to mathematical evaluation; and an unwavering faith that if the world just used more computers, and used them properly, social problems would disappear and we'd create a digitally enabled utopia. It's not true. There has never been, nor will there ever be, a technological innovation that moves us away from the essential problems of human nature. Why, then, do people persist in thinking there's a sunny technological future just around the corner?

I started thinking about technochauvinism one day when I was talking with a twenty-something friend who works as a data scientist. I mentioned something about Philadelphia schools that didn't have enough books.

"Why not just use laptops or iPads and get electronic textbooks?" asked my friend. "Doesn't technology make everything faster, cheaper, and better?"

He got an earful. (You'll get one too in a later chapter.) However, his assumption stuck with me. My friend thought that technology was always the answer. I thought technology was only appropriate if it was the right tool for the task.

Somehow, in the past two decades, many of us began to assume that computers get it right and humans get it wrong. We started saying things like "Computers are better because they are more objective than people." Computers have become so pervasive in every aspect of our lives that when something goes awry in the machine, we assume that it's our fault, rather than assume something went wrong within the thousands of lines of code that make up the average computer program. In reality, as any software developer can tell you, the problem is usually in the machine somewhere. It's probably in poorly designed or tested code, cheap hardware, or a profound misunderstanding of how the actual users would use the system.

If you're anything like my data scientist friend, you're probably skeptical. Maybe you're a person who loves your cell phone, or maybe you've been told your whole life that computers are the wave of the future. I hear you. I was told that too. What I ask is that you stick with me as I tell some stories about people who built technology, then use these stories to think critically about the technology we have and the people who made it. This isn't a technical manual or a textbook; it's a collection of stories with a purpose. I chose a handful of adventures in computer programming, each of which I undertook in order to understand something fundamental about technology and contemporary tech culture. All of those projects link together in a sort of chain, building an argument against technochauvinism. Along the way, I'll explain how some computer technology works and unpack the human systems that technology serves.

The first four chapters of the book cover a few basics about how computers work and how computer programs are constructed. If you already are crystal-clear on how hardware and software work together, or you already know how to write code, you'll probably breeze through chapters 1–3 on computation and go quickly to chapter 4, which focuses on data. These first chapters are important because all artificial intelligence (AI) is built on the same foundation of code, data, binary, and electrical impulses. Understanding what is real and what is imaginary in AI is crucial. Artificial superintelligences, like on the TV show *Person of Interest* or *Star Trek*, are imaginary. Yes, they're fun to imagine, and it can inspire wonderful creativity to think about the possibilities of robot domination and so on—but they aren't real. This book hews closely to the real mathematical, cognitive, and computational concepts that are in the actual academic discipline of artificial intelligence: knowledge representation and reasoning, logic, machine learning, natural language processing, search, planning, mechanics, and ethics.

In the first computational adventure (chapter 5), I investigate why, after two decades of education reform, schools still can't get students to pass standardized tests. It's not the students' or the teachers' fault. The problem is far bigger: the companies that create the most important state and local exams also publish textbooks that contain many of the answers, but low-income school districts can't afford to buy the books.

I discovered this thorny situation by building artificial intelligence software to enable my reporting. Robot reporters have been in the news in recent years because the Associated Press (AP) is using bots to write routine business and sports stories. My software wasn't inside a robot (it didn't need to be, although I'm not averse to the idea), nor did it write any stories for me (ditto). Instead, it was a brand-new application of old-school artificial intelligence that helped reveal some fascinating insights. One of the most surprising findings of this computational investigation was that, even in our high-tech world, the simplest solution—a book in the hands of a child—was quite effective. It made me wonder why we are spending so much money to put technology into classrooms when we already have a cheap, effective solution that works well.

The next chapter (chapter 6) is a whirlwind tour through the history of machines, specifically focused on Marvin Minsky—commonly known as the father of artificial intelligence—and the enormous role that 1960s counterculture played in developing the beliefs about the Internet that exist in 2017, the time this book was written. My goal here is to show you how the dreams and goals of specific individuals have shaped scientific knowledge, culture, business rhetoric, and even the legal framework of today's technology through deliberate choices. The reason we don't have national territories on the Internet, for example, is that many of the people who made the Internet believed they could make a new world beyond government—much like they tried (and failed) to make new worlds in communes.

In thinking about tech, it's important to keep another cultural touchstone in mind: Hollywood. A great deal of what people dream about making in tech is shaped by the images they see in movies, TV programs, and books. (Remember my childhood robot?) When computer scientists refer to *artificial intelligence*, we make a distinction between general AI and narrow AI. *General AI* is the Hollywood version. This is the kind of AI that would power the robot butler, might theoretically become sentient and take over the government, could result in a real-life Arnold Schwarzenegger as the Terminator, and all of the other dread possibilities. Most computer scientists have a thorough grounding in science fiction literature and movies, and we're almost always happy to talk through the hypothetical possibilities of general AI.

Inside the computer science community, people gave up on general AI in the 1990s.[3] General AI is now called *Good Old-Fashioned Artificial Intelligence* (GOFAI). *Narrow AI* is what we actually have. Narrow AI is purely mathematical. It's less exciting than GOFAI, but it works surprisingly well and we can do a variety of interesting things with it. However, the linguistic confusion is significant. Machine learning, a popular form of AI, is not GOFAI. Machine learning is narrow AI. The name is confusing. Even to me, the phrase *machine learning* still suggests there is a sentient being in the computer.

The important distinction is this: general AI is what we want, what we hope for, and what we imagine (minus the evil robot overlords of golden-age science fiction). Narrow AI is what we have. It's the difference between dreams and reality.

Next, in chapter 7, I define machine learning and demonstrate how to "do" machine learning by predicting which passengers survived the *Titanic* crash. This definition is necessary for understanding the fourth project (chapter 8), in which I ride in a self-driving car and explain why a self-driving school bus is guaranteed to crash. The first time I rode in a self-driving car was in 2007, and the computerized "driver" almost killed me in a Boeing parking lot. The technology has come a long way since then, but it still fundamentally doesn't work as well as a human brain. The cyborg future is not coming anytime soon. I look at our fantasies about technology replacing humans and explore why it's so hard to admit when technology isn't as effective as we want it to be.

Chapter 9 is a springboard for exploring why *popular* is not the same as *good* and how this confusion—which is perpetuated by machine-learning techniques—is potentially dangerous. Chapters 10 and 11 are also programming adventures, in which I start a pizza-calculating company on a cross-country hackathon bus trip (it's popular but not good) and try to repair the US campaign finance system by building AI software for the 2016 presidential election (it's good but not popular). In both cases, I build software that works—but it doesn't work as expected. Its demise is instructive.

My goal in this book is to empower people around technology. I want people to understand how computers work so that they don't have to be intimidated by software. We've all been in that position at one time or another. We've all felt helpless and frustrated in the face of a simple task that should be easy, but somehow isn't because of the technological interface. Even my students, who grew up being called digital natives, often find the digital world confusing, intimidating, and poorly designed.

When we rely exclusively on computation for answers to complex social issues, we are relying on artificial unintelligence. To be clear: it's the computer that's artificially unintelligent, not the person. The computer doesn't give a flying fig about what it does or what you do. It executes commands to the best of its abilities, then it waits for the next command. It has no sentience, and it has no soul.

People are always intelligent. However, smart and well-intentioned people act like technochauvinists when they are blind to the faults of computational decision making or they are excessively attached to the idea of using computers to the point at which they want to use computers for everything—including things for which the computer is not suited.

I think we can do better. Once we understand how computers work, we can begin to demand better quality in technology. We can demand systems that *truly* make things cheaper, faster, and better instead of putting up with systems that promise improvement but in fact make things unnecessarily complicated. We can learn to make better decisions about the downstream effects of technology so that we don't cause unintentional harm inside complex social systems. And we can feel empowered to say "no" to technology when it's not necessary so that we can live better, more connected lives and enjoy the many ways tech can and does enhance our world.