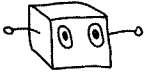


YOU LOOK  
LIKE A THING  
 AND  
I LOVE YOU

How Artificial Intelligence Works  
and Why It's Making the World  
a Weirder Place

Janelle Shane



**VORACIOUS**

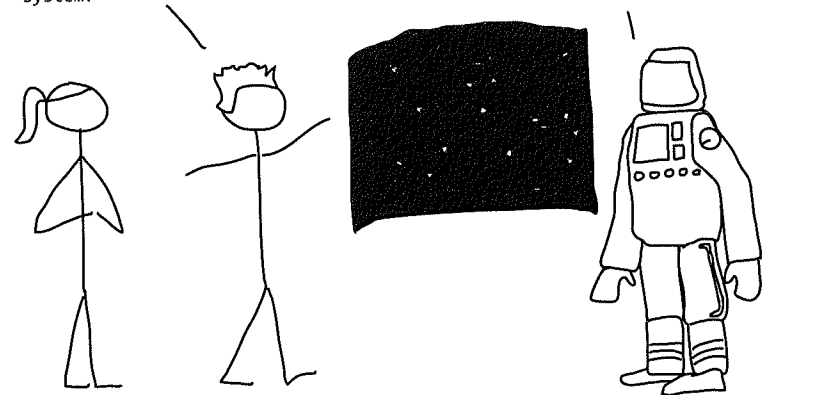
Little, Brown and Company

New York ■ Boston ■ London

## CHAPTER 1

# What Is AI?

Quick, AI! Calculate warp coordinates for the Bal Panda system!



Oops. Wrong kind of AI.  
I'm just a guy in a robot suit.  
This is awkward.

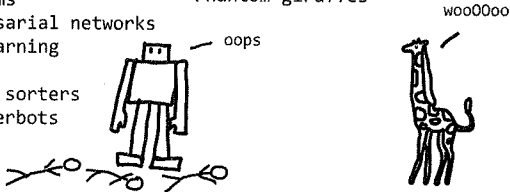
If it seems like AI is everywhere, it's partly because "artificial intelligence" means lots of things, depending on whether you're reading science fiction or selling a new app or doing academic research. When someone says they have an AI-powered chatbot, should we expect it to have opinions and feelings like the fictional C-3PO? Or is it just an algorithm that learned to guess how humans are likely to respond to a given phrase? Or a spreadsheet that matches words in your question with a library of preformulated answers? Or an underpaid human who types all the answers from some remote location? Or — even — a completely scripted conversation where human and AI

are reading human-written lines like characters in a play? Confusingly, at various times, all these have been referred to as AI.

For the purposes of this book, I'll use the term AI the way it's mostly used by programmers today: to refer to a particular style of computer program called a machine learning algorithm. This chart shows a bunch of the terms I'll be covering in this book and where they fall according to this definition.

Things called AI

Called AI in this book	In this book, but not AI
Machine learning algorithms	Science fiction AIs
Deep learning	Rules-based programs
Neural networks	Humans in robot costumes
Recurrent neural networks	Robots reading scripts
Markov chains	Humans hired to pretend to be AIs
Random forests	Sentient cockroaches
Genetic algorithms	Phantom giraffes
Generative adversarial networks	
Reinforcement learning	
Predictive text	
Magical sandwich sorters	
Unfortunate murderbots	



Everything that I'm calling "AI" in this book is also a machine learning algorithm — let's talk about what that is.

**KNOCK, KNOCK, WHO'S THERE?**

To spot an AI in the wild, it's important to know the difference between machine learning algorithms (what we're calling AI in this book) and traditional (what programmers call rules-based) programs. If you've ever done basic programming, or even used HTML to design a website, you've

used a rules-based program. You create a list of commands, or rules, in a language the computer can understand, and the computer does exactly what you say. To solve a problem with a rules-based program, you have to know every step required to complete the program's task and how to describe each one of those steps.

But a machine learning algorithm figures out the rules for itself via trial and error, gauging its success on goals the programmer has specified. The goal could be a list of examples to imitate, a game score to increase, or anything else. As the AI tries to reach this goal, it can discover rules and correlations that the programmer didn't even know existed. Programming an AI is almost more like teaching a child than programming a computer.

**Rules-based programming**

Let's say I wanted to use the more familiar rules-based programming to teach a computer to tell knock-knock jokes. The first thing I'd do is figure out all the rules. I'd analyze the structure of knock-knock jokes and discover that there's an underlying formula, as follows:

```

Knock, knock.
Who's there?
[Name]
[Name] who?
[Name] [Punchline]

```

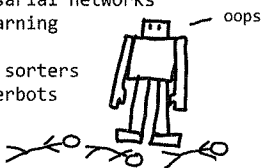
Once I set this formula in stone, there are only two slots free that the program can control: [Name] and [Punchline]. Now the problem is reduced to just generating these two items. But I still need rules for generating them.

I could set up a list of valid names and a list of valid punchlines, as follows:

are reading human-written lines like characters in a play? Confusingly, at various times, all these have been referred to as AI.

For the purposes of this book, I'll use the term *AI* the way it's mostly used by programmers today: to refer to a particular style of computer program called a machine learning algorithm. This chart shows a bunch of the terms I'll be covering in this book and where they fall according to this definition.

Things called AI	
Called AI in this book	In this book, but not AI
Machine learning algorithms	Science fiction AIs
Deep learning	Rules-based programs
Neural networks	Humans in robot costumes
Recurrent neural networks	Robots reading scripts
Markov chains	Humans hired to pretend to be AIs
Random forests	Sentient cockroaches
Genetic algorithms	Phantom giraffes
Generative adversarial networks	
Reinforcement learning	
Predictive text	
Magical sandwich sorters	
Unfortunate murderbots	



Everything that I'm calling "AI" in this book is also a machine learning algorithm — let's talk about what that is.

## KNOCK, KNOCK, WHO'S THERE?

To spot an AI in the wild, it's important to know the difference between **machine learning algorithms** (what we're calling AI in this book) and traditional (what programmers call **rules-based**) programs. If you've ever done basic programming, or even used HTML to design a website, you've

used a rules-based program. You create a list of commands, or rules, in a language the computer can understand, and the computer does exactly what you say. To solve a problem with a rules-based program, you have to know every step required to complete the program's task and how to describe each one of those steps.

But a machine learning algorithm figures out the rules for itself via trial and error, gauging its success on goals the programmer has specified. The goal could be a list of examples to imitate, a game score to increase, or anything else. As the AI tries to reach this goal, it can discover rules and correlations that the programmer didn't even know existed. Programming an AI is almost more like teaching a child than programming a computer.

## Rules-based programming

Let's say I wanted to use the more familiar rules-based programming to teach a computer to tell knock-knock jokes. The first thing I'd do is figure out all the rules. I'd analyze the structure of knock-knock jokes and discover that there's an underlying formula, as follows:

```
Knock, knock.
Who's there?
[Name]
[Name] who?
[Name] [Punchline]
```

Once I set this formula in stone, there are only two slots free that the program can control: [Name] and [Punchline]. Now the problem is reduced to just generating these two items. But I still need rules for generating them.

I could set up a list of valid names and a list of valid punchlines, as follows:

Names	Punchlines
Lettuce	in, it's cold out here!
Harry	up, it's cold out here!
Dozen	anybody want to let me in?
Orange	you going to let me in?

Now the computer can produce knock-knock jokes by choosing a name-punchline pair from the list and slotting it into the template. This doesn't create *new* knock-knock jokes but only gives me jokes I already know. I might try making things interesting by allowing [it's cold out here!] to be replaced with a few different phrases: [I'm being attacked by eels!] and [lest I transform into an unspeakable eldritch horror]. Then the program can generate a new joke:

Knock, knock.  
 Who's there?  
 Harry.  
 Harry who?  
 Harry up, I'm being attacked by eels!

I could replace [eels] with [an angry bee] or [a manta ray] or any number of things. Then I can get the computer to generate even more new jokes. With enough rules, I could potentially generate hundreds of jokes.

Depending on the level of sophistication I'm going for, I could spend a lot of time coming up with more advanced rules. I could find a list of existing puns and figure out a way to transform them into punchline format. I could even try programming in pronunciation rules, rhymes, semihomophones, cultural references, and so forth in an attempt to get the computer to recombine them into interesting puns. If I'm clever about it, I can get

the program to generate new puns that nobody's ever thought of. (Although one person who tried this discovered that the algorithm's list of sayings contained words and phrases that were so old or obscure that almost nobody could understand its jokes.) No matter how sophisticated my joke-making rules get, though, I'm still telling the computer exactly how to solve the problem.

### Training AI

But when I train AI to tell knock-knock jokes, I don't make the rules. The AI has to figure out those rules on its own.

The only thing I give it is a set of existing knock-knock jokes and instructions that are essentially, "Here are some jokes; try to make more of these." And the materials I give it to work with? A bucket of random letters and punctuation.

Then I leave to get coffee.

The AI gets to work.

The first thing it does is try to guess a few letters of a few knock-knock jokes. It's guessing 100 percent randomly at this point, so this first guess could be anything. Let's say it guesses something like "qasdnw,m sne?mso d." As far as it knows, this is how you tell a knock-knock joke.

Then the AI looks at what those knock-knock jokes are *actually* supposed to be. Chances are it's very wrong. "All right," says the AI, and it subtly adjusts its own structure so that it will guess a little more accurately next time. There's a limit to how drastically it can change itself, because we don't want it to try to memorize every new chunk of text it sees. But with a minimum of tweaking, the AI can discover that if it guesses nothing but *k*'s and spaces, it will at least be right some of the time. After looking at one batch of knock-knock jokes and making one round of corrections, its idea of a knock-knock joke looks something like this:

k k k k k  
 kk k kkkok  
 k kkkk  
 k  
  
 kk  
 kk k kk  
  
 keokk k  
  
 k  
 k

Now, it's not the world's greatest knock-knock joke. But with this as a starting point, the AI can move on to a second batch of knock-knock jokes, then another. Each time, it adjusts its joke formula to improve its guesses.

After a few more rounds of guessing and self-adjusting, it has learned more rules. It has learned to employ the occasional question mark at the end of a line. It is beginning to learn to use vowels (o in particular). It's even making an attempt at using apostrophes.

noo,  
 Lnoc noo  
 Kor?  
 hnos h nc  
 pt'b oa to'  
  
 asutWtnl  
 toy nooc  
 doc kKe

w ce  
 e

How well did its rules about generating knock-knock jokes match reality? It still seems to be missing something.

If it wants to get closer to generating a passable knock-knock joke, it's going to have to figure out some rules about which *order* the letters come in. Again, it starts by guessing. The guess that o is always followed by q? Not so great, it turns out. But then it guesses that o is often followed by ck. Gold. It has made some progress. Behold its idea of the perfect joke:

Whock  
 Whock  
 Whock  
 Whock  
 Whock Whock Whock  
 Whock Whock  
 Whock  
 Whock

It's not quite a knock-knock joke — it sounds more like some kind of chicken. The AI's going to need to figure out some more rules.

It looks at the dataset again. It tries new ways to use "ock," looking for new combinations that match its example knock-knock jokes better.

nooc hock hork  
 aKoo kWhoc  
 Khock  
 hors Whnes Whee  
 noo ooc  
 Koo?

loo  
 noc no'c Who h orea  
 ao no oo k KeockWnork Koos  
 Thoe  
 nock K ock  
 Koc  
 hoo  
 KKnock Knock Thock Whonk

All this progress happens in just a few minutes. By the time I return with my coffee, the AI has *already* discovered that starting with “Knock Knock / Who’s There?” fits the existing knock-knock jokes really, really well. It decides to apply this rule *always*.

But the rest of the formula takes quite a while to figure out, and the AI frequently devolves into playing a game of telephone with itself—or acting out a robot version of the “Who’s on First?” comedy routine:

Knock Knock  
 Who’s There?  
 Iane  
 Aatar who?  
 Aaane who?  
 Aan who?  
 Anac who?  
 Iobe who?  
 Irata who?

Soon it figures out the rest of the formula, but no actual, you know, *words*:

Knock Knock  
 Who’s There?

Bool  
 Hane who?  
 Scheres are then the there

It also sometimes gets a bit carried away with the length of the punchline.

Knock Knock  
 Who’s There?  
 Cheed  
 I?len who?  
 Ita gee the lasd emotory oud ane weonges oak anet ane  
 tee soore theis onithit the where your thasg tate  
 tine if thele ter the theung!

At some point in its training, it learns the joke about the Cow with No Lips, which becomes its Favorite Thing Ever. If you’re not familiar, the joke about the Cow with No Lips goes something like:

Knock, knock.  
 Who’s there?  
 A cow with no lips.  
 A cow with no lips who?  
 A cow with no lips says ooo ooo ooooooo.

According to the AI, this is comedy gold. So the AI spends a while thinking that the punchline to Every Single Joke is “ooo ooo ooooooo”...

Knock Knock  
 Who’s There?  
 Alec  
 Anic who?

Wow co poo the tho tho door to the soout oo ooo oooo  
 ooo oo oo oo oo ooo oo oo oo ooo ooo oooo ooo ooo oo  
 oo ooo ooo ooo ooo ooo oo oo ooo ooo ooo ooo oo oo  
 oo ooo ooo ooo ooo . . .

It has discovered a rule, all right. Eventually it (reluctantly, I assume) tones down how often it uses the Cow with No Lips rule to make punch-lines. But it still has problems that are remarkably similar to two people telling knock-knock jokes over a very bad phone connection, maybe forgetting who was actually telling the joke:

Knock Knock  
 Who's There?  
 Alther  
 Water who?  
 Water who who?  
 Who who?  
 Arton and out on the moor?

At last it figures out the structure of a knock-knock joke and proceeds to make jokes that, while technically correct, don't make all that much sense. Many of them are partially plagiarized from jokes in the training dataset.

Knock Knock  
 Who's There?  
 Robin  
 Robin who?  
 Robin you! Git me and I'm leaving

Knock Knock  
 Who's There?

Bet  
 Beef who?  
 Beef ano with no lips aslly.

Ireland  
 Ireland who?  
 Ireland you money, butt.

And then. It produced. An actual joke. That it had composed entirely on its own, without plagiarizing from the dataset, and that was not only intelligible but also actually...funny?

Knock Knock  
 Who's There?  
 Alec  
 Alec who?  
 Alec- Knock Knock jokes.

Did the AI suddenly begin to understand knock-knock jokes and English-language puns? Almost definitely not, given the very small size of the dataset. But the freedom that the AI had — free rein over the entire set of possible characters — allowed it to try new combinations of sounds, one of which ended up actually working. So more of a victory for the infinite monkey theory\* than a proof of concept for the next AI-only comedy club.

---

\* The old adage that a monkey writing randomly on a typewriter for an infinite amount of time will eventually produce the entire works of Shakespeare actually pretty accurately describes the "brute force" method of searching for a solution to a problem by systematically trying everything. Ideally, using AI to solve the problem is an improvement over this. Ideally.



The beauty of letting AI make its own rules is that a single approach—here's the data; you try to figure out how to copy it—works on a lot of different problems. If I had given the joke-telling algorithm another dataset instead of knock-knock jokes, it would have learned to copy that dataset instead.

It could create new species of birds:

Yucatan Jungle-Duck  
 Boat-billed Sunbird  
 Western Prong-billed Flowerpecker  
 Black-capped Flufftail  
 Iceland Reedhaunter  
 Snowy Mourning Heron-Robin

Or new perfumes:

Fancy Ten  
 Eau de Boffe  
 Frogant Flower  
 Momite  
 Santa for Women

Or even new recipes.

#### **BASIC CLAM FROSTING**

main dish, soups

1 lb chicken  
 1 lb pork, cubed  
 ½ clove garlic, crushed  
 1 cup celery, sliced  
 1 head (about ½ cup)

6 tablespoon electric mixer  
 1 teaspoon black pepper  
 1 onion—chopped  
 3 cup beef broth the owinls for a fruit  
 1 freshly crushed half and half; worth water

With pureed lemon juice and lemon slices in a 3-quart pan.

Add vegetables, add chicken to sauce, mixing well in onion. Add bay leaf, red pepper, and slowly cover and simmer covered for 3 hours. Add potatoes and carrots to simmering. Heat until sauce boils. Serve with pies.

If the liced pieces cooked up desserts, and cook over wok.

Refrigerate up to ½ hour decorated.

Yield: 6 servings

#### **JUST LET THE AI FIGURE IT OUT**

Given a set of knock-knock jokes and no further instruction, the AI was able to discover a lot of the rules that I would have otherwise had to manually program into it. Some of its rules I would never have thought to program in or wouldn't even have known existed—such as The Cow with No Lips Is the Best Joke.

This is part of what makes AIs attractive problem solvers, and is particularly handy if the rules are really complicated or just plain mysterious. For example, AI is often used for image recognition, a surprisingly complicated

task that's difficult to do with an ordinary computer program. Although most of us are easily able to identify a cat in a picture, it's really hard to come up with the rules that define a cat. Do we tell the program that a cat has two eyes, a nose, two ears, and a tail? That also describes a mouse and a giraffe. And what if the cat is curled up or facing away? Even writing down the rules for detecting a single eye is tricky. But an AI can look at tens of thousands of images of cats and come up with rules that correctly identify a cat most of the time.

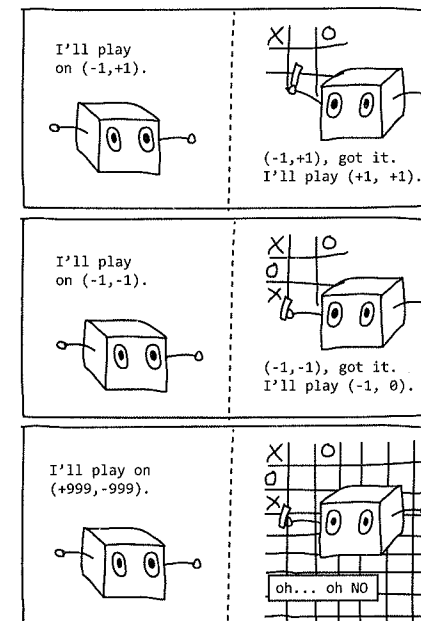
Sometimes AI is only a small part of a program while the rest of it is rules-based scripting. Consider a program that lets customers call their banks for account information. The voice-recognition AI matches spoken sounds to options in the help-line menu, but programmer-issued rules govern the list of options the caller can access and the code that identifies the account as belonging to the customer.

Other programs start out as AI-powered but switch control over to humans if things get tough, an approach called pseudo-AI. Some customer-service chat windows work like this. When you begin a conversation with a bot, if you act too confusing, or if the AI detects that you are getting annoyed, you may suddenly find yourself chatting with a human instead. (A human who unfortunately now has to deal with a confused and/or annoyed customer—maybe a “talk to a human” option would be better for customer *and* employee.) Today's self-driving cars work this way, too—the driver has to always be ready to take control if the AI gets flustered.

AI is also great at strategy games like chess, for which we know how to describe all possible moves but not how to write a formula that tells us what the best next move is. In chess, the sheer number of possible moves

and complexity of game play means that even a grandmaster would be unable to come up with hard-and-fast rules governing the best move in any given situation. But an algorithm can play a bunch of practice games against itself—millions of them, more than even the most dedicated grandmaster—to come up with rules that help it win. And since the AI learned without explicit instruction, sometimes its strategies are very unconventional. Sometimes a little *too* unconventional.

If you don't tell AI which moves are valid, it may find and exploit strange loopholes that completely break your game. For example, in 1997 some programmers built algorithms that could play tic-tac-toe remotely against each other on an infinitely large board. One programmer, rather than designing a rules-based strategy, built an AI that could evolve its own approach. Surprisingly, the AI suddenly began winning all its games. It turned out that the AI's strategy was to place its move very, very far away, so that when its



opponent's computer tried to simulate the new, greatly expanded board, the effort would cause it to run out of memory and crash, forfeiting the game.<sup>1</sup> Most AI programmers have stories like this—times when their algorithms surprised them by coming up with solutions they hadn't expected. Sometimes these new solutions are ingenious, and sometimes they're a problem.

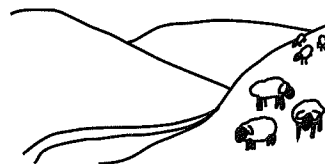
At its most basic, all AI needs is a goal and a set of data to learn from and it's off to the races, whether the goal is to copy examples of loan decisions a human made or predict whether a customer will buy a certain sock or maximize the score in a video game or maximize the distance a robot can travel. In every scenario, AI uses trial and error to invent rules that will help it reach its goal.

### SOMETIMES ITS RULES ARE BAD

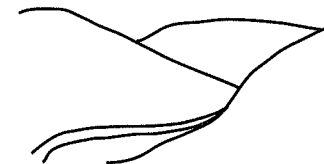
Sometimes, an AI's brilliant problem-solving rules actually rely on mistaken assumptions. For example, some of my weirdest AI experiments have involved Microsoft's image recognition product, which allows you to submit any image for the AI to tag and caption. Generally, this algorithm gets things right—identifying clouds, subway trains, and even a kid doing some sweet skateboarding tricks. But one day I noticed something odd about its results: it was tagging sheep in pictures that definitely did not contain any sheep. When I investigated further, I discovered that it tended to see sheep in landscapes that had lush green fields—whether or not the sheep were actually there. Why the persistent—and specific—error? Maybe during training this AI had mostly been shown sheep that were in fields of this sort and had failed to realize that the “sheep” caption referred to the animals, not to the grassy landscape. In other words, the AI had been looking at the wrong thing. And sure enough, when I showed it examples of sheep that were *not* in lush green fields, it tended to get confused. If I showed it pictures of sheep in cars, it would tend to label them as dogs or

cats instead. Sheep in living rooms also got labeled as dogs and cats, as did sheep held in people's arms. And sheep on leashes were identified as dogs. The AI also had similar problems with goats—when they climbed into trees, as they sometimes do, the algorithm thought they were giraffes (and another similar algorithm called them birds).

A herd of sheep grazing on a lush green landscape



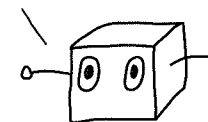
A herd of sheep grazing on a lush green landscape



Although I couldn't know for sure, I could guess that the AI had come up with rules like Green Grass = Sheep, and Fur in Cars or Kitchens = Cats. These rules had served it well in training but failed when it encountered the real world and its dizzying variety of sheep-related situations.



...fluffy bird?



Training errors like these are common with image recognition AIs. But the consequences of these mistakes can be serious. A team at Stanford University once trained an AI to tell the difference between pictures of healthy skin and pictures of skin cancer. After the researchers trained their AI, however, they discovered that they had inadvertently trained a ruler detector instead—many of the tumors in their training data had been photographed next to rulers for scale.<sup>2</sup>

## HOW TO DETECT A BAD RULE

It's often not that easy to tell when AIs make mistakes. Since we don't write their rules, they come up with their own, and they don't write them down or explain them the way a human would. Instead, the AIs make complex interdependent adjustments to their own internal structures, turning a generic framework into something that's fine-tuned for an individual task. It's like starting with a kitchen full of generic ingredients and ending with cookies. The rules might be stored in the connections between virtual brain cells or in the genes of a virtual organism. The rules might be complex, spread out, and weirdly entangled with one another. Studying an AI's internal structure can be a lot like studying the brain or an ecosystem — and you don't need to be a neuroscientist or an ecologist to know how complex those can be.

Researchers are working on finding out just how AIs make decisions, but in general, it's hard to discover what an AI's internal rules actually are. Often it's merely because the rules are hard to understand, but at other times, especially when it comes to commercial and/or government algorithms, it's because the algorithm itself is proprietary. So unfortunately, problems often turn up in the algorithm's results when it's already in use, sometimes making decisions that can affect lives and potentially cause real harm.

For example, an AI that was being used to recommend which prisoners would be paroled was found to be making prejudiced decisions, unknowingly copying the racist behaviors it found in its training.<sup>3</sup> Even without understanding what bias is, AI can still manage to be biased. After all, many AIs learn by copying humans. The question they're answering is not "What is the best solution?" but "What would the humans have done?"

Systematically testing for bias can help catch some of these common problems before they do damage. But another piece of the puzzle is learning to anticipate problems before they occur and designing AIs to avoid them.

## FOUR SIGNS OF AI DOOM

When people think of AI disaster, they think of AIs refusing orders, deciding that their best interests lie in killing all humans, or creating terminator bots. But all those disaster scenarios assume a level of critical thinking and a humanlike understanding of the world that AIs won't be capable of for the foreseeable future. As leading machine learning researcher Andrew Ng put it, worrying about an AI takeover is like worrying about overcrowding on Mars.<sup>4</sup>

That's not to say that today's AIs can't cause problems. From slightly annoying their programmers all the way to perpetuating prejudices or crashing a self-driving car, today's AIs are not exactly harmless. But by knowing a little about AI, we can see some of these problems coming.

Here's how an AI disaster might actually play out today.

Let's say a Silicon Valley startup is offering to save companies time by screening job candidates, identifying the likely top performers by analyzing short video interviews. This could be attractive — companies spend a lot of time and resources interviewing dozens of candidates just to find that one good match. Software never gets tired and never gets hangry, and it doesn't hold personal grudges. But what are the warning signs that what the company is building is actually an AI disaster?

### *Warning sign number 1: The Problem Is Too Hard*

The thing about hiring good people is that it's really difficult. Even humans have trouble identifying good candidates. Is this candidate genuinely excited to work here or just a good actor? Have we accounted for disability or differences in culture? When you add AI to the mix, it gets even more difficult. It's nearly impossible for AI to understand the nuances of jokes or tone or cultural references. And what if a candidate makes a reference to the day's current events? If the AI was trained on data collected the

previous year, it won't have a chance of understanding—and it might punish the candidate for saying something it finds nonsensical. To do the job well, the AI will have to have a huge range of skills and keep track of a large amount of information. If it isn't capable of doing the job well, we're in for some kind of failure.

***Warning sign number 2: The Problem Is Not What We Thought It Was***

The problem with designing an AI to screen candidates for us: we aren't really asking the AI to identify the best candidates. We're asking it to identify the candidates that most resemble the ones our human hiring managers liked in the past.

That might be okay if the human hiring managers made great decisions. But most US companies have a diversity problem, particularly among managers and particularly in the way that hiring managers evaluate resumes and interview candidates. All else being equal, resumes with white-male-sounding names are more likely to get interviews than those with female-and/or minority-sounding names.<sup>5</sup> Even hiring managers who are female and/or members of a minority themselves tend to unconsciously favor white male candidates.

Plenty of bad and/or outright harmful AI programs are designed by people who thought they were designing an AI to solve a problem but were unknowingly training it to do something entirely different.

***Warning sign number 3: There Are Sneaky Shortcuts***

Remember the skin-cancer-detecting AI that was really a ruler detector? Identifying the minute differences between healthy cells and cancer cells is difficult, so the AI found it a lot easier to look for the presence of a ruler in the picture.

If you give a job-candidate-screening AI biased data to learn from

(which you almost certainly did, unless you did a lot of work to scrub bias from the data), then you also give it a convenient shortcut to improve its accuracy at predicting the “best” candidate: prefer white men. That's a lot easier than analyzing the nuances of a candidate's choice of wording. Or perhaps the AI will find and exploit another unfortunate shortcut—maybe we filmed our successful candidates using a single camera, and the AI learns to read the camera metadata and select only candidates who were filmed with that camera.

AIs take sneaky shortcuts all the time—they just don't know any better!

***Warning sign number 4: The AI Tried to Learn from Flawed Data***

There's an old computer-science saying: garbage in, garbage out. If the AI's goal is to imitate humans who make flawed decisions, perfect success would be to imitate those decisions exactly, flaws and all.

Flawed data, whether it's flawed examples to learn from or a flawed simulation with weird physics, will throw an AI for a loop or send it off in the wrong direction. Since in many cases our example data is the problem we're giving the AI to solve, it's no wonder that bad data leads to a bad solution. In fact, warning signs numbers 1 through 3 are most often evidence of problems with data.

**DOOM — OR DELIGHT**

The job-candidate-screening example is, unfortunately, not hypothetical. Multiple companies already offer AI-powered resume-screening or video-interview-screening services, and few offer information about what they've done to address bias or to account for disability or cultural differences or to find out what information their AIs use in the screening process. With

careful work, it's at least possible to build a job-candidate-screening AI that is measurably less biased than human hiring managers — but without published stats to prove it, we can be pretty sure that bias is still there.

The difference between successful AI problem solving and failure usually has a lot to do with the suitability of the task for an AI solution. And there are plenty of tasks for which AI solutions are more efficient than human solutions. What are they, and what makes AI so good at them? Let's take a look.