SWIPE TO UNLOCK

A Primer on Technology and Business Strategy

Neel Mehta
Parth Detroja
Aditya Agashe

Swipe to Unlock

Published by Belle Applications, Inc.
swipetounlock.com

2nd edition, 2st revision (September 2018)

ISBN-13: 978-1976182198
ISBN-10: 1976182190

To my friends and family, for supporting me no matter how crazy my dreams get — Neel

To my friends and family for their never-ending support, and Professor Deborah Streeter for believing in my vision, which helped make this book possible — Parth

To my family and friends, thanks for supporting my passion for business and helping me push past my fears to embrace entrepreneurship — Adi

When you grow up, you tend to get told that the world is the way it is and your life is just to live your life inside the world, try not to bash into the walls too much, try to have a nice family, have fun, save a little money. That's a very limited life. Life can be much broader, once you discover one simple fact, and that is that everything around you that you call life was made up by people that were no smarter than you. And you can change it, you can influence it, you can build your own things that other people will use… That's maybe the most important thing. It's to shake off this erroneous notion that life is there and you're just gonna live in it, versus embrace it, change it, improve it, make your mark upon it… Once you learn that, you'll never be the same again.

— Steve Jobs
(who, by the way, never wrote any code for Apple)

Contents

Chapter 1:
Introduction

Understanding technology is essential in today's world, no matter your career. Doctors are using artificial intelligence to diagnose patients.[1] Finance, payments, and banking have all been moving online.[2] Businesspeople are adapting to a world where, as of 2018, the world's five biggest companies are all tech companies.[3] Even farmers are using drones to grow better crops.[4] Whether you're a doctor, banker, farmer, or most anything else, you need to know how technology works.

If you don't know how to code, you might be a little worried right now.

But here's the thing: you don't need to know how to code to understand technology. We believe that all the most important technology topics — from how the internet works to how apps like Snapchat make money — can be explained in plain, simple English. You'll find explanations of these topics, and many more, in *Swipe to Unlock*.

In this book, we've distilled all the big ideas we've learned from working around the tech industry, from tiny startups to massive corporations. Inside *Swipe to Unlock*, you'll find breakdowns of the hottest technologies, translations of the

most mysterious tech buzzwords, and analyses of the business strategies that power the tech industry. We think it's everything you need to know about technology and the business strategy behind it, and we think absolutely anyone can learn it.

But enough about us. How about you?

Who this book is for

The goal of *Swipe to Unlock* isn't to teach you how to code. It's to teach you how to think like a technologist. Whenever you come across a technology topic at work or on the news, you'll be able to think through how the technology works, why it was made that way, how it makes money, and how it might help or harm people. While specific apps and companies are always coming and going, the core concepts you'll pick up in *Swipe to Unlock* should prove useful for a long time to come.

If you want a business, marketing, product management, or other non-engineering role at a tech company, you need to understand the fundamentals of technology. In *Swipe to Unlock*, we'll teach you how to think like a software engineer. You'll learn how software is made, how the internet works, and what buzzwords like "big data," "machine learning," and "APIs" mean. (Don't worry if you've never heard of these terms — we'll break them down throughout the book!)

If you're already a software developer but want to move more toward product management roles, we'll teach you the business strategy insights you need to know. We'll pose a variety of real-world business questions and work through the answers. *Why does Amazon offer free shipping with Prime if it loses them money? Why did Facebook acquire Instagram? How do software companies make money when most apps are free?* You'll also learn how to break down complex technical concepts to laypeople. Questions like "explain cloud computing to a five-year-old" are surprisingly common in tech interviews — so we'll show you some useful analogies, like how cloud computing is similar to calling an Uber.

If you work outside the tech industry, you'll learn about the digital tools you need to stay ahead of the curve. We'll teach you what jargon like "predictive analytics" and "the cloud" means, and we'll introduce you to companies that have been using these technologies to boost sales, cut costs, and improve efficiency.

Even if you don't need to know any technology for your career, you still use it every day — you probably have a very advanced piece of technology in your pocket (or your hand) right now. *Swipe to Unlock* will make you a better-educated consumer by answering some of the questions you might have had while following the news or using your favorite apps. *What caused the fake news epidemic on social media? What's this "net neutrality" debate that everyone's talking about? How much of my personal data does Google own?* We'll talk about these questions and many others.

No matter why you're reading this book, we think you'll find plenty of valuable insights and ideas, and you'll learn how to think — and speak! — like a technologist.

But before we start, let's dive a bit deeper into the topics you'll learn about.

What's inside

We'll kick off *Swipe to Unlock* by looking under the hood of the technology you use every day, from your phone's operating system (like Android or iOS) to your favorite apps. In the middle chapters, we'll break down important concepts like "the cloud," "big data," and hacking. Then we'll zoom out and look at some of the business, legal, and political implications of technology. We'll finish up with a look at emerging trends in technology and some predictions for the future.

All three of us hate reading textbooks, so we didn't write *Swipe to Unlock* like a textbook. Instead, every single section is in the form of a question, where we step through real-world examples. *How does Spotify recommend songs to you? How do self-driving cars work? How could a hacker hold your computer for "ransom"?* We'll break down many cases like these.

Along the way, you'll read about some surprising, and maybe even funny, stories. *How did Target figure out that a teenager was pregnant before her own father did? How did a single typo take down 20% of the internet?* We'll even meet some interesting characters, like a Wall Street trader who blasted right through the Allegheny Mountains in his quest to make a perfectly straight internet cable.

A final note: we couldn't possibly cover everything in full detail in *Swipe to Unlock*, so we'll give you dozens — and, in some cases, hundreds — of citations per chapter. You can use these links to dive deeper into any topic that interests you.

For job hunters

If you're looking for a non-developer role at a tech company, we're glad you're reading *Swipe to Unlock* — we think it'll be useful! A big reason we wrote this book was because there just weren't any comprehensive resources for non-experts to learn the basics of technology.

Swipe to Unlock aims to teach you the technology and business concepts you'll need to know, and it'll help you understand tech companies' favorite jargon. We think the technical topics will be especially useful if you're coming from a non-technical background, and the business concepts will especially help if you're coming from an engineering background.

Bear in mind, though, that *Swipe to Unlock* isn't designed to give you resume or networking advice, and it won't teach you how to code or prepare for interviews. For that kind of information, check out the Resources section of our website at swipetounlock.com/resources, where we share links to some useful books and articles.

The questions we explore in *Swipe to Unlock* aren't designed to resemble actual interview questions; instead, they'll give you the technical and business insight to craft answers that'll set you apart from the pack. Let us explain with a couple of examples.

One example: we'll talk about how Google's algorithm decides which ads to show to users. Google probably wouldn't ask you to explain this algorithm at an interview, but they could ask you how you would increase ad revenue from a particular group of

users. Knowing how the targeting algorithm works is crucial to creating a strong answer.

A second example from the business world: we'll discuss why Microsoft acquired LinkedIn. Microsoft interviewers probably won't ask you to explain the reasons for the purchase, but they could ask you how you'd improve Microsoft Office for businesses. In your answer, you could draw on our discussion of the purchase, where we discuss how Microsoft and LinkedIn data could work together. You could mention one interesting idea we'll cover: analyzing a company's current employees via LinkedIn to see what kinds of talent they need to add. We think answers like these can help you stand out from the crowd.

Third, we'll give you some great anecdotes to take to the interview. For instance, we'll introduce you to Robinhood, an app that lets you trade stocks without paying brokers any commission. Robinhood has a clever way of making money: it earns interest on the unspent money sitting in users' accounts. At an interview, what if someone asks you to come up with a new money-making strategy for Venmo, an app that lets you send money to friends for free? Well, you could pull out the Robinhood story and propose that Venmo earn interest on money sitting in Venmo users' accounts. We think our stories will help you find many creative ideas like this.

In short, *Swipe to Unlock* is an essential complement to books full of sample interview questions. Sample questions alone won't help you understand the tech industry, think strategically about software products, or become fluent in technology jargon, but we believe *Swipe to Unlock* will. Best of luck!

Who we are

Before we jump into the main content of *Swipe to Unlock*, let us introduce ourselves. The three of us met while working at Microsoft. We quickly realized that Silicon Valley does a poor job making the world of technology accessible to non-experts. We believe that everyone can — and should! — understand the fundamentals of technology, and we're passionate about helping make that happen. That's why we wrote *Swipe to Unlock*.

Here's a bit more about us.

Neel Mehta is a Product Manager at Google. He has previously worked at Microsoft and graduated *cum laude* from Harvard University.

Parth Detroja is a Product Manager at Facebook. He has previously worked at Microsoft, Amazon, and IBM. He graduated *summa cum laude* from Cornell University.

Aditya Agashe is a Product Manager at Microsoft. Previously, he was the founder and CEO of Belle Applications. He graduated *cum laude* from Cornell University.

Thank you and good luck!

Whatever your personal or career goals, we hope *Swipe to Unlock* helps you. If you'd like more tips along the way, learn more and join our mailing list at swipetounlock.com. We regularly release new interviews, articles, and posts to help non-developers understand the world of technology.

Thank you again for choosing to read *Swipe to Unlock*, and we hope you enjoy!

Neel Mehta
hathix.com
linkedin.com/in/neelmehta18

Parth Detroja
parthdetroja.com
linkedin.com/in/parthdetroja

Aditya Agashe
whoisadi.com
linkedin.com/in/adityaagashe

Chapter 2:
Software Development

Let's start our tour of the world of technology by taking a closer look at the apps you use every day.

Every piece of software, from the original Pac-Man game to the latest version of Snapchat, is made of instructions, or code, that people wrote down and that computers follow. While the code behind apps like Spotify or Yelp might seem like magical spells, many of the core ideas are simple. We won't get into the nitty-gritty of code in this chapter, but we'll take a high-level overview of three major building blocks of apps and websites.

First, software is built around algorithms, or well-defined procedures that computers use to solve a problem. For instance, the algorithm to find the area of a triangle is to multiply the base and the height, then cut the result in half. There are algorithms to predict the weather, recommend movies to watch, process credit card payments, and more. We'll take a look under the hood of some famous apps and websites to reveal the algorithms that make them tick.

Then we'll turn to a second building block: APIs, or application programming interfaces, which let apps borrow algorithms and data from other apps. We'll explore how APIs

work and why developers use them, and we'll learn how to analyze an app to figure out what APIs it uses.

Algorithms and APIs help developers build out software, but to perfect it, developers turn to a third tool: A/B tests. In an A/B test, you show different versions of a feature to different groups of users: for instance, half the users get a red button while the other half get a blue button. Then you look at certain statistics, or "metrics," to see which variation performed better, and you push the winning version to all users. We'll finish up this chapter with a look at how some apps and websites use this powerful, scientific technique to improve their software.

Once you understand these three building blocks, you can start peering under the hood of any app or website to learn how it works. Let's take a look.

How does Google search work?

Whenever you search on Google, the search engine combs through the over 30 trillion webpages on the internet and finds the top 10 results for your question.[1] 92% of the time, you'll click on a result on the first page (that is, among the top 10 results).[2] Finding the top 10 things out of 30 trillion is really hard — it's about as hard as trying to find a penny randomly dropped somewhere in New York City.[3] Yet Google does this expertly and in just half a second, on average.[4] But how?

Google doesn't actually visit every page on the internet every time you ask it something. Google actually stores information about webpages in databases (tables of information, like in Excel), and it uses algorithms that read those databases to decide what to show you. Let's dive into how these work.

First, Google needs to build a database of every webpage on the internet. Google uses programs called spiders to "crawl" over webpages until it's found all of them (or at least, what Google thinks is all of them). The spiders start on a few webpages and add those to Google's list of pages, called the "index." Then, the spiders follow all the outgoing links on those pages and find a new set of pages, which they add to the index. Next, they follow all the links on *those* pages, and so on, until Google can't find anything else.

This process is always ongoing; Google is always adding new pages to its index or updating pages when they change. The index is huge, weighing in at over 100 million gigabytes.[5] If you tried to fit that on one-terabyte external hard drives, you'd need

100,000 — which, if you stacked them up, would be around a mile high.[6]

Word search

When you search Google, it grabs your query (the text you typed into the search bar) and looks through its index to find the webpages that are most relevant.

How might Google do this? The simplest way would be to just look for occurrences of a particular keyword, kind of like hitting Ctrl+F or Cmd+F to search a giant Word document. Indeed, this is how search engines in the 90's used to work: they'd search for your query in their index and show the pages that had the most matches,[7] an attribute called keyword density.[8]

This turned out to be pretty easy to game. If you searched for the candy bar Snickers, you'd imagine that snickers.com would be the first hit. But if a search engine just counted the number of times the word "snickers" appeared on a webpage, anyone could make a random page that just said "snickers snickers snickers snickers" (and so on) and jump to the top of the search results. Clearly, that's not very useful.

PageRank

Instead of keyword density, Google's core innovation is an algorithm called PageRank, which its founders Larry Page and Sergey Brin created for their PhD thesis in 1998.[9] Page and Brin noticed that you can estimate a webpage's importance by looking at which other important pages link to them.[10] It's like how, at a party, you know someone is popular when they're

surrounded by *other* popular people. PageRank gives each webpage a score that's based on the PageRank scores of every other page that links to that page.[11] (The scores of *those* pages depend on the pages that link to them, and so on… this stops eventually thanks to some math tricks from linear algebra.[12])

For instance, if we made a brand-new webpage about Abraham Lincoln, it would initially have very low PageRank. If some obscure blog added a link to our page, our page would get a small boost to its PageRank. PageRank cares more about the quality of incoming links rather than the quantity,[13] so even if dozens of obscure blogs linked to our page, we wouldn't gain much. But if a New York Times article (which would probably have a high PageRank) linked to our page, our page would get a huge PageRank boost.

Once Google finds all the pages in its index that mention your search query, it ranks them using several criteria, including PageRank.[14] Google considers many smaller criteria as well: it considers how recently a webpage was updated, ignores websites that look spammy (like the "snickers snickers snickers snickers" site we mentioned earlier), considers your location (it could return the NFL if you search for "football" in the US, but the English Premier League if you search for "football" in England), and more.[15]

One particularly interesting technique is called "query rewriting," where Google looks for words with similar meanings to the word you searched for. For instance, if you searched for "sofa", Google could also return webpages that include terms like "couch," "chair," "seat," or "furniture."[16]

Gaming Google?

There are pitfalls to PageRank, however. Much like spammers abused keyword density (as with "snickers snickers snickers snickers"), spammers have now started making "link farms," or webpages that contain tons of unrelated links. Website owners can pay link farms to include a link to their webpages, which would artificially boost their PageRank.[17] However, Google has gotten pretty good at catching and ignoring link farms.[18]

There are some more mainstream ways to game Google, though. An entire industry, called search engine optimization (SEO), has sprung up to help website owners crack Google's search algorithm and make sure their webpages appear at the top of Google searches.[19] The most basic form of SEO is getting more pages to link to your page. SEO includes plenty more techniques, such as putting the right keywords in your page's title and headings or making all of your site's pages link to each other.[20]

Constant changes

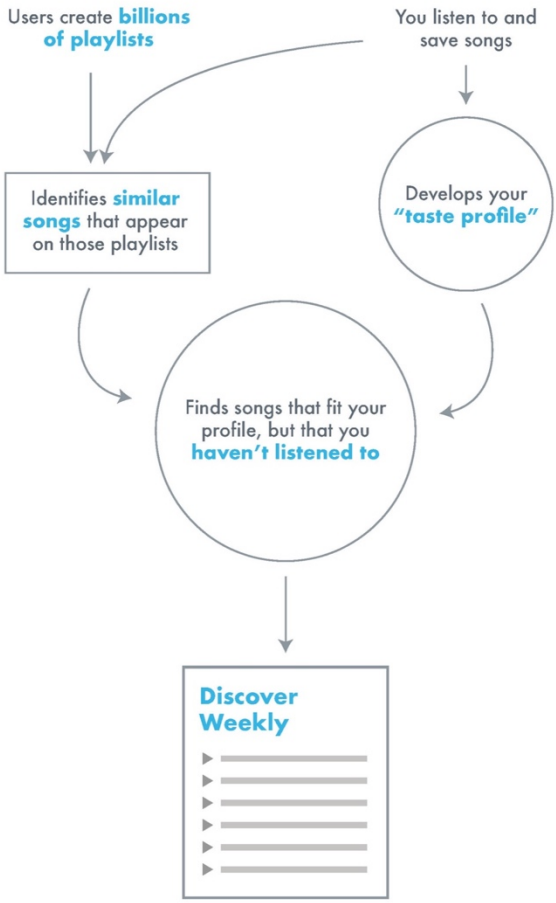
One final thing to note: while PageRank has always remained at the core of Google, the search engine is always changing. Google made 1,600 changes to its search algorithm in 2016 alone,[21] and in recent years it's added features like driving directions and "infoboxes," or cards that show quick information about your search term from Wikipedia.

Infoboxes are a new feature of Google's search results. They grab relevant information about your search from sources like Wikipedia. Source: Google search

How does Spotify recommend songs to you?

Every Monday morning, Spotify sends its listeners a playlist of 30 songs that are, almost magically, perfectly tuned to their tastes. This playlist, called Discover Weekly, has been a hit: within six months of launching in June 2015, it was streamed over 1.7 billion times.[22] But how does Spotify get to know its 75 million users so intimately?

Spotify does employ music experts who hand-curate public playlists,[23] but there's no way they could do that for all 75 million Spotify users. Instead, Spotify has turned to a computer algorithm they run every week.[24]



Spotify's algorithm for automatically recommending songs for you. Source: Quartz[25]

The Discover Weekly algorithm starts by looking at two basic pieces of information. First, it looks at all the songs you've listened to and liked enough to add to your library or playlists. It's even smart enough to know that if you skipped a song within the first 30 seconds, you probably didn't like it! Second, it looks at all the playlists that other people have made, with

the assumption that every playlist has some thematic connection; for instance, you could have a "running" playlist or a "Beatles jam" playlist.[26]

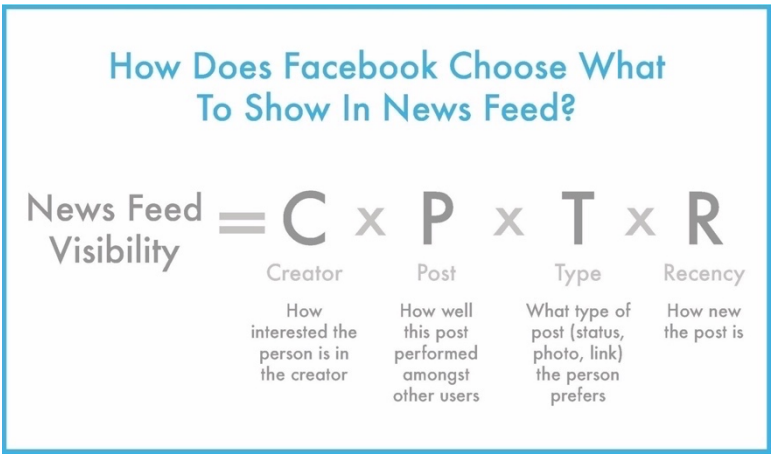
Once it has this data, Spotify uses two methods to find songs you might like. The first method involves comparing the two datasets to figure out which new songs are related to the ones you like. For instance, suppose someone made a playlist with eight songs on it, and seven of those are in your library. You probably like this type of song, so Discover Weekly might recommend the one song that isn't in your library.[27] This technique is called "collaborative filtering," and it's the same technique that Amazon uses to suggest items you might like based on your purchase history and the purchases of millions of other users.[28]

The second method that Spotify uses to make your playlist is your "taste profile." Based on just the songs you've listened to and liked, Spotify will determine which genres (e.g. indie rock or R&B) and micro-genres (e.g. chamber pop or New Americana) you like and recommend songs from those genres. It's just a different form of their strategy for recommending songs based on past listening patterns.[29]

Recommendation systems are getting more and more popular across the app landscape: besides Spotify, Netflix suggests movies, YouTube suggests videos, Facebook suggests friends, and so on. Once you understand how Spotify runs its recommendation system, the others all become easier to understand, because they largely use the same methods, like collaborative filtering.[30]

How does Facebook decide what shows up in your news feed?

Over a billion people look at their Facebook news feed every day, and Americans spend almost as much time on Facebook as they do interacting face-to-face with other people.[31] Because it draws so many eyeballs, the news feed has tremendous power. The news feed can influence our mood, put us in an ideological echo chambers,[32] or even influence who we're going to vote for.[33] In short, what appears in your news feed matters. So how does Facebook decide what to show in your news feed?



A simplified explanation of Facebook's news feed algorithm. Source: TechCrunch[34]

More specifically, how does Facebook take the hundreds (or thousands) of updates you get every day and sort them for you? Like Google, it uses an algorithm to figure out what's most important. There are about 100,000 personalized factors, but we'll focus on four key ones.[35]

The first factor is who posted it. Facebook will show you more posts from people you've interacted with more (e.g. people you've messaged more or tagged more), with the assumption that you're more likely to engage with their future posts.[36]

Second is the post's quality. The more people have engaged (e.g. liked or commented) with a post, the more interesting Facebook thinks it is, so the more likely it is to appear at the top of your news feed.[37]

Third, the type of post. Facebook figures out what kinds of posts (videos, articles, photos, etc.) you interact with a lot and shows you more of those.[38]

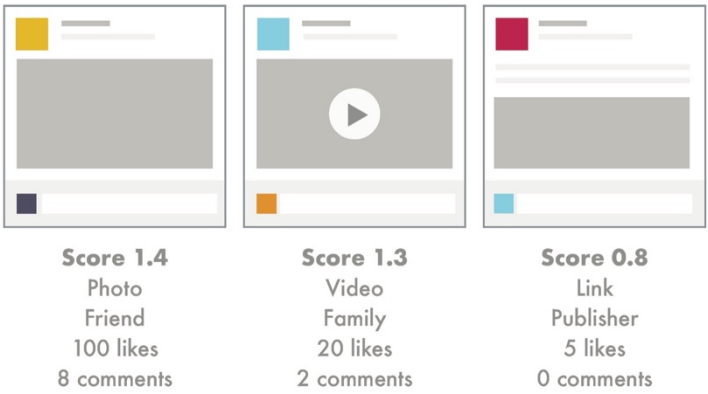
The fourth major factor is "recency": newer stories get ranked higher.[39]

There are plenty more factors, though. TIME found a few:

> Use a phone with a slow mobile connection and you may see less video. Writing 'congratulations' in a comment signals the post is probably about a big life event, so it will get a boost. Liking an article after you clicked it is a stronger positive signal than liking before, since it means you probably read the piece and enjoyed it.[40]

As you can tell, Facebook really tries to maximize the probability that you're going to like or comment on the posts in your news feed, a metric called engagement. After all, the more you like your news feed, the more you're going to scroll

down, and the more you scroll down, the more ads you'll see. Ads, of course, are how Facebook makes most of its money.[41]



An example of how Facebook ranks posts and determines what appears on your news feed. Source: TechCrunch[42]

Fighting fake news

Algorithms like Facebook's news feed algorithm are incredibly powerful, but the danger is that they're still easy for crafty hackers to game. With no human oversight, the algorithms could be turned against us.

A famous example is the fake news epidemic that swept Facebook during the 2016 American presidential election.[43] Recall that the news feed algorithm doesn't consider how true or reputable a post is; it only cares about maximizing engagement.[44] Fake newsmongers took advantage of this to attack politicians they didn't like, posting outrageous and demonstrably false news articles around Facebook. These articles naturally drew many clicks and comments, so

Facebook's news feed algorithm propelled them to the top of many people's news feeds.[45]

How could we fix weaknesses like this in the news feed algorithm? We might need to bring back a human touch to the algorithm — which is a bit ironic because algorithms are designed to reduce the amount of work humans have to do in the first place. For instance, Facebook and Google introduced features that let people flag fake news posts, which their updated algorithms can then hide.[46]

Even outside the battle against fake news, Facebook has been bringing in humans to improve the news feed algorithm, because counting likes can only tell you so much about how people feel about posts. In 2015, Facebook started hiring focus groups that scroll through their news feeds and give feedback to the people behind Facebook's news feed algorithm.[47] (Yes, that's right, you *can* be a professional Facebook tester, but we recommend staying in school regardless.)

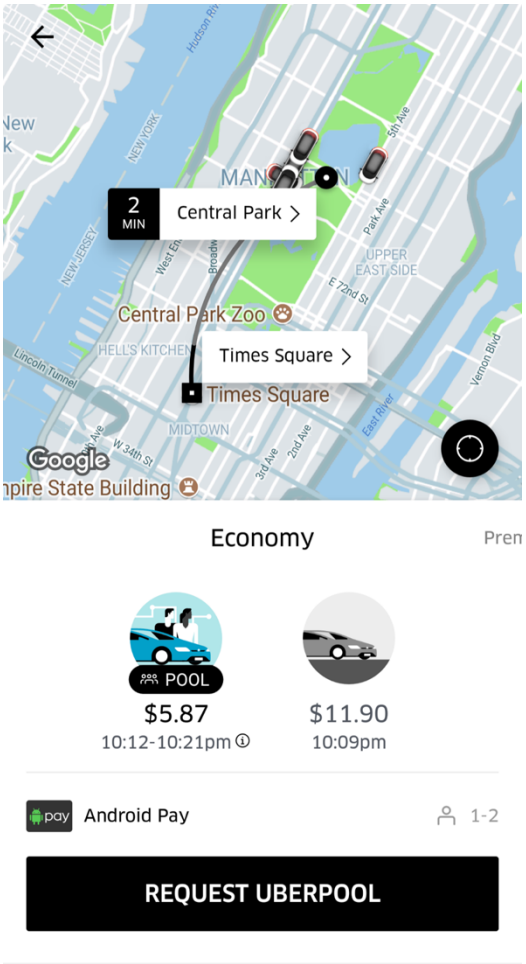
Algorithms aren't magic spells that run the world. They're just sets of rules (though complex ones) that other people wrote to make computers do a particular task. And, as Facebook shows, sometimes machines and people need to work together.

What technologies do Uber, Yelp, and Pokémon Go all have in common?

Ever tried making a digital map like on Google Maps? It's pretty hard. You need to keep track of every road, building, city, and shoreline on the planet. Maybe you'll need a fleet of cars to drive around the world and take pictures and measurements, as Google has done for Google Maps.[48] Oh, you'll also need to make a nice interface with features like panning and zooming. (Don't get us started on making an algorithm to find driving directions between two points.)

In short, making a digital map is really hard. But apps like Uber, Pokémon Go, and Yelp need to include a map to show where available cars are, help the user find wild Pokémon, or display nearby restaurants. Do these apps need to spend thousands of person-hours making their own maps?

Fortunately for them, the answer is no. And if you've ever used these apps, you can probably figure out what they do instead: they embed a Google Map in their app. Searching up restaurants? Yelp drops pins on a Google Map centered on your location. Want to Uber downtown? The app draws the route you'll take on a Google Map and calculates the approximate time it'll take to get there.[49]



Uber uses the Google Maps API to draw a map of your area and predict your car's travel time. Source: Uber on Android

Google lets you include a small snippet of code in your app to draw a Google Map. It also provides other snippets of code to let you draw on the maps, calculate driving directions between points on the map, and even find out the speed limit for a particular road. All these tools are cheap or even free.[50] These tools are big win for developers; they can use the technology

that has taken Google years to perfect with just a small amount of code. There's no need to reinvent the wheel!

These snippets of code that let you borrow another app's functionality or data are called APIs, or application programming interfaces. In short, APIs let apps talk to each other. Let's look at three main kinds of APIs.

The first kind of API lets one app ask another specialized app to solve a particular problem, like calculating driving directions, sending text messages, or translating sentences. It's like how you could call a plumber or carpenter to fix problems around your house instead of trying to do it yourself. One example of this kind of API: processing credit card payments is pretty difficult, so apps that charge you (like Uber[51]) can use PayPal's Braintree API, which lets anyone use PayPal's algorithm for processing credit card payments with just a few lines of code.[52] Similarly, it's a pain to write code that sends emails or text messages — so when apps like Venmo need to send you confirmation emails or texts, they can just use a specialized API.[53] And any app that lets you log in via Facebook or Google uses those websites' "Single Sign-On" APIs for verifying someone's identity.[54]

The second type of API lets one app ask another app to hand over some interesting information, such as sports scores, recent Tweets, or today's weather. It's like calling the front desk at a hotel to learn which museums and restaurants they recommend. ESPN offers an API that lets you get rosters for every major-league sports team and scores for every game.[55] New York's subway system lets you track where trains are and predict when the next train will arrive at a station.[56] The US

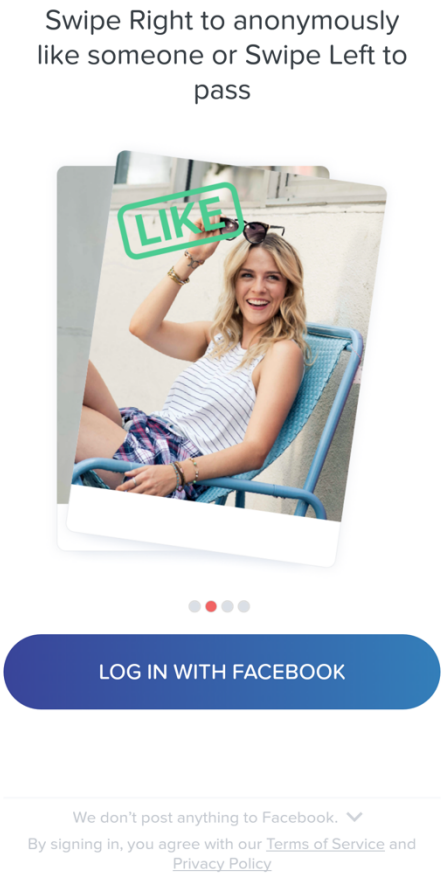
government offers APIs that let you access all kinds of data about American healthcare, energy, agriculture, and more.[57] And don't worry, there's even an API to get random images of cats.[58]

The final kind of APIs lets developers access features of the device itself. Snapchat taps into your phone's Camera API to zoom, focus, and snap photos. Google Maps itself uses your phone's Geolocation API to figure out where in the world you are. Your phone even has sensors called accelerometers and gyroscopes, which fitness apps (and Pokémon Go) use to determine which direction you're walking and how fast you're moving.[59]

It's worth noting that APIs aren't perfect. Using an API makes app developers' lives easier, but it also makes their apps dependent on the API.[60] If PayPal's Braintree payment API somehow stopped working, Uber would be unable to process payments — not great for their business. Still, a specialized company's API will probably be more reliable than any similar feature that a non-specialist company could make themselves.

All this brings us back to the question: what technologies do Uber, Yelp, and Pokémon Go have in common? They all use APIs, especially the Google Maps API, to avoid reinventing the wheel. Indeed, APIs are a core part of pretty much every app out there.

Why does Tinder make you log in with Facebook?



Tinder on Android. Note that the only way to sign in is with Facebook.

If you've ever used the dating app Tinder, you'll notice that they make you log in with your Facebook account before you can set up a profile. Once you connect your Facebook profile, Tinder imports information like your profile picture, your age, your list of friends, and the Facebook pages you like.[61] As you

might have guessed, this is done through an API that Facebook offers. With this API, any app can let users create accounts by linking their Facebook profiles.[62]

Why does Tinder use this API? For one, it ensures that there are no empty profiles (which no one would want to swipe on) since some basic information always gets imported from Facebook.[63] Second, requiring Facebook login helps them stop bots, since bots are less likely to have Facebook accounts.[64] Third, this helps Tinder make better matches: by gathering your friend list, Tinder can show you how many mutual friends you have with each potential match, and that sense of connection probably encourages people to swipe more. Finally, by getting the Facebook profiles of all their users, Tinder can get some deep insights into their user base, such as how old they are, where they live, or what they're interested in.[65] These insights can help Tinder tweak their app's design or advertising strategies.

Signing up with Facebook is also helpful for users. Making your profile with Tinder becomes much faster when most of your basic information and pictures are already imported from Facebook.[66] Seeing more completed profiles and fewer bots improves your experience, as well.[67] And signing up with Facebook means you don't have to remember yet another username and password.[68]

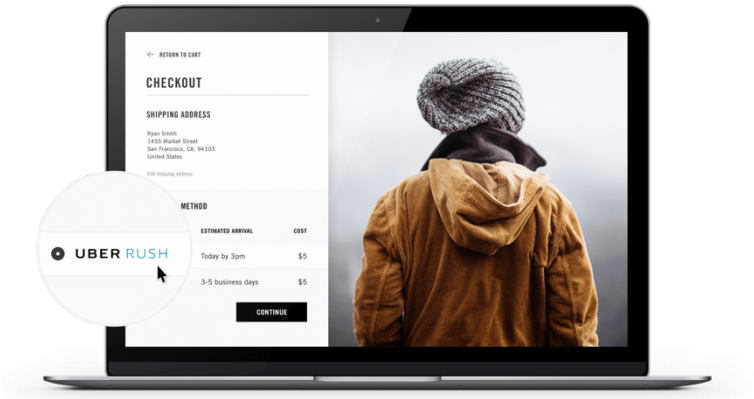
Why would Facebook publish the API that lets people sign in to other websites using their Facebook credentials? Well, when you use Facebook' sign-in API (often called "single sign-on," or SSO) to register for Tinder, Facebook realizes that you're a Tinder user. Facebook gets similar data points when you use

Facebook to log into other websites. Facebook can then use this data to target ads at you more effectively, for instance by showing more dating-related ads to Tinder users.[69]

As this example shows, publishing APIs is a great way for companies to gain data and usage, and using APIs helps apps save time and offer better features. The end goal is to help you, the user.

Why does Uber let other apps use Ubers to deliver their products?

In 2015, Uber published an API called UberRUSH that lets other app developers ship items short distances using Ubers.[70] For instance, a local florist could use the UberRUSH API to deliver flowers using Ubers instead of hiring their own fleet of drivers.[71] In other words, the API lets other companies rent Uber's huge driver network and logistics infrastructure.[72] Why did Uber bother going through the hassle of making and publishing this API?



An example of a website that uses the UberRUSH API to deliver items using Ubers. Source: Uber[73]

The most obvious reason is money. Uber can charge a bit for companies to use UberRUSH.[74] There are some more interesting reasons, though.

The second reason is to help Uber expand its market. Uber is already a leader in transporting people on demand,[75] but before UberRUSH, Uber wasn't competing in the market of

transporting *things* on demand. Uber wanted to be sure they won that market before someone else, like the fast food-delivery service Postmates, beat them to it.[76] In fact, UberRUSH might even let Uber start competing with traditional shipping companies like FedEx and UPS in the same-day shipping space.[77] It's a savvy business move.

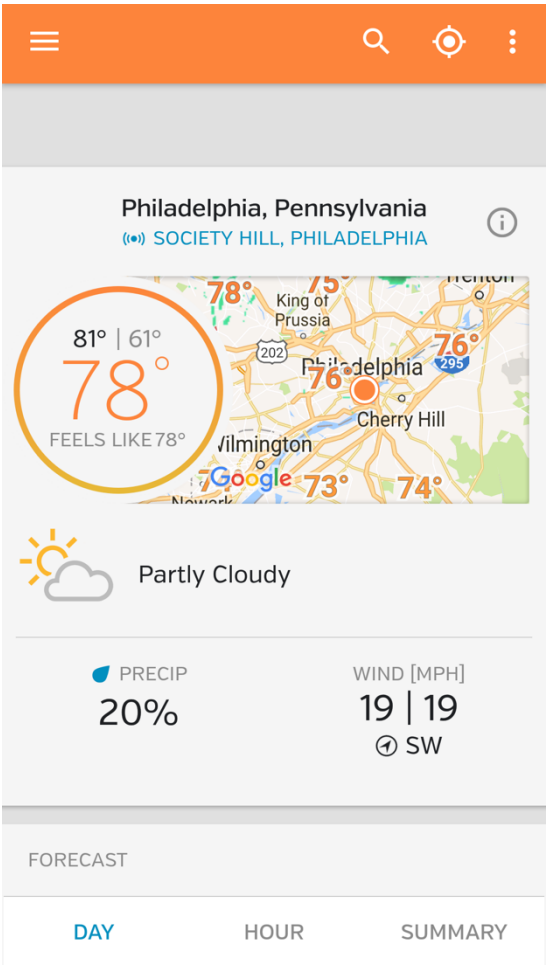
The third reason is users and data. By expanding into the "ship things short distances on demand" market, Uber can attract new customers and drivers and run more trips. Uber's key strength (or "core competency," as businesspeople call it[78]) is its huge driver network,[79] so attracting new drivers is a big win. More trips also means more ride data, which can help Uber improve its services — for instance, by helping Uber figure out where to place cars to minimize riders' wait times.[80]

So why does Uber publish an API that lets other app developers deliver stuff with Ubers? It's not just to be nice. The API lets Uber make money, expand into new markets, and gather more customers, drivers, and data. Meanwhile, the companies that use UberRUSH to ship things avoid the cost and headache of operating their own driver network.[81] It's a win-win.

How does your weather app work?

Pull out your phone and open the weather app. Let's use it as an example to show how you can understand almost any app by analyzing which APIs and algorithms it uses.

Your weather app will probably look something like this:



The Weather Underground app on Android.

Let's dive into the screenshot to figure out how the app works.

First, consider the APIs it uses. As you can tell from the map, the Weather Underground app embeds a Google Map (just like Uber, Yelp, and so many other apps) and then uses the Google Maps API to draw temperatures and weather patterns (green blobs for rain, white blobs for snow, etc.) on top.[82] Second, whenever you plug in a zip code, the app uses a zip code API to figure out which city you mean.[83] For instance, entering 60601 shows you the weather for Chicago, Illinois. Third, if you hit the target in the top right, the app will use your phone's Geolocation API to figure out your GPS location and show you the weather for that place. And finally, Weather Underground gets its weather data, such as the current temperature and probability of rain, from the National Weather Service, which offers all the weather data that the American government collects through a free API.[84]

Current weather information is great, but weather app users really want to know about future weather. So forecasters like the National Weather Service and Weather Underground use algorithms to predict future weather based on current conditions, recent weather trends, and other factors. The NWS, for instance, uses the Weather Forecasting Model, an algorithm that simulates the physics of the atmosphere to make weather predictions.[85]

These APIs and algorithms are the heart of the weather app, and they might just be enough to build a weather app yourself. If you use this same lens to look at your favorite apps — from

Pokémon Go to Snapchat to your to-do list app — you can start making sense of how these apps really work.

Why does every Washington Post article have two versions of the same headline?

Take a look at the following two screenshots of the same Washington Post piece. Notice anything different?

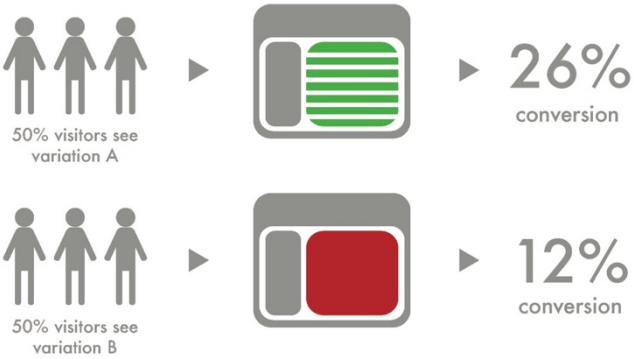
Notice a difference? The Washington Post shows people different versions of the same headline. It's part of an experiment to figure out which one gets more clicks. Source: The Washington Post[86]

That's right, the headlines are different! In 2016, the Washington Post rolled out a feature that lets article writers specify two different headlines for every article.[87] But why?

It's actually an experiment that the newspaper runs to maximize the number of clicks on its articles.[88] This experiment automatically shows one version of the headline to one group of visitors; let's say one-half of them, randomly chosen. It shows the other version to the remaining visitors. After letting the experiment run for a while, developers look at particular statistics, or metrics, like the number of clicks on the headline. Developers decide which version is better and show the winning version to everyone. This is a simple but powerful

way to improve an app's metrics. For instance, the first version of the above headline was clicked 3.3% of the time, while the bottom version was clicked 3.9%[89] — that's an 18% jump, just by changing a few words!

This technique is called A/B testing, which you might remember from the start of the chapter. A/B testing is a powerful, data-driven way to improve online products.[90] It's called "A/B testing" because you compare at least two versions of a feature, A and B.



A/B testing shows at least two variations of the same feature (A and B) and compares relevant metrics to decide which variation to push to all users. In this case, everyone would start seeing variation A, which was better at getting users to take a desired action (or "conversion"). Source: VWO[91]

Not sure which marketing catchphrase will get people to buy? Instead of endless debating, just run an A/B test! Not sure if a red or green "sign up" button will get more people to click? Run a test! (If you're curious, the red button got 34% more clicks in one experiment.[92]) Not sure which Tinder profile picture will get you the most swipes? Guess what: Tinder

automatically runs A/B tests to decide which of your photos, when shown as your main profile picture, gets you the most right swipes.[93]

That brings us back to the question: why does every Washington Post article have two versions of the same headline? It's part of the Washington Post's A/B testing framework, called Bandito. Bandito tries out different versions of the headline to see which one people click on more, and then shows the winning headline more often.[94] (Technically, it's a little more sophisticated than an A/B test — it's a so-called "multi-armed bandit" algorithm[95] — but the core idea is the same.)

A/B testing is very popular among news outlets. BuzzFeed uses A/B tests to find the most clickbaity headlines as well.[96] A BuzzFeed competitor called Upworthy actually tries up to 25 versions of the same headline in its quest to find the perfect one.[97] A/B testing is very important: according to Upworthy, the difference between a decent headline and a perfected one is 1,000 vs. 1,000,000 views.[98]

Many more apps and websites use A/B testing. Facebook, for instance, is always rolling out new features to "limited test audiences."[99] Snapchat lets advertisers A/B test their ads, to pick the ones that get tapped the most.[100] Even brick-and-mortar stores can do A/B testing: one startup, for instance, lets stores vary the background music they play to maximize shoppers' spending.[101] A/B testing really is everywhere.

Significance testing

There's one important caveat to keep in mind whenever you're doing statistical tests: you need to check whether your observed findings happened because of something meaningful or were just due to chance. For instance, if you flip a coin six times and get five heads, you can't be confident that the coin is unfairly weighted toward heads — it could just be dumb luck. But if you flipped the coin six hundred times and got five hundred heads, you might be on to something.

When companies perform A/B tests, experimenters report how one version changed a particular metric compared to the other version. They also report a statistic called a *p*-value, which shows the probability that the difference they observed was due to chance.[102] Usually, if $p < 0.05$ (i.e. there's a less than 5% chance that the difference was just random), they can assume the change was meaningful, or "statistically significant."[103] Otherwise, they can't be sure that their results weren't just dumb luck.

For example, say Amazon made the "Add to Cart" buttons a bit bigger for half their users and found a 5% increase in sales, with $p = 0.15$. While the bigger button seems like a great move, there's a 15% chance that the sales boost was caused by dumb luck, not the button. That 0.15 is greater than 0.05, so Amazon's testers won't roll out the bigger button. (A cutoff of 0.05 might seem too strict, but statisticians are very careful people.[104])

So if you ever get clickbaited by a headline like "18 Food Arguments So Strong That They End Friendships,"[105] don't feel too bad — you're up against a powerful blend of social

science, software development, and statistics. Like it or not, A/B testing is extremely effective.

Chapter 12:
Trends Going Forward

Congratulations! By this point, you've learned how to think like a technologist. You understand topics ranging from the cloud to big data to internet security. You can explain how today's most popular technology works, and you have a strong understanding of the technology industry.

That's great, but you need to know more. It's not enough to only know about the past and present. The world of technology is always changing, so to stay ahead, you need to learn about current trends and predictions about where the tech world will be going.

So, in this final chapter, we'll explore some of the most important trends you need to know going forward. We'll start by looking at technologies that would have seemed like science fiction in the 20th century: cars that can drive themselves, robots that can hold conversations with you, and software that can make fake audio and video good enough to fool humans. While we're at it, we'll explain popular buzzwords like "machine learning," "artificial intelligence," "neural networks," and "natural language processing." You might have heard technologists, pundits, or entrepreneurs raving about

these technologies — we'll help you understand what they're talking about and why it matters.

Along the way, we'll tackle some of the age-old questions about technology, which are becoming more and more relevant as computers get ever smarter. Can computers think like people? And, most important of all, can they replace us?

Finally, we'll investigate the future of the tech industry. Will the tech titan Amazon become worth $1 trillion, or will it get broken up? And what does that mean for the industry?

We'll answer all these big questions and more in this chapter. Ready for one last round? Let's go.

How do self-driving cars work?

Google started a buzz when it began testing prototypes of its self-driving cars on the streets of Mountain View, California in 2015.[1] People have been excited about self-driving cars, with one expert arguing that they could cut accident rates by 90%.[2] Others have feared that the self-driving car industry could destroy entire industries, like America's trucking industry, which employs 3.5 million drivers.[3] Because of all this potential impact, people all over the world want to know: how exactly do these cars work? And when (if ever) will they hit the streets worldwide? Let's take a look under the hood.

No humans need apply

Let's start by exploring how a car can drive itself. Self-driving cars use a combination of sensors attached to the car, a computer in the trunk, and a company's servers in the cloud.[4]

First, the car needs to know where in the world it is. For this, the car uses an onboard GPS to figure out its rough location.[5] The GPS isn't always perfectly accurate, though, so the car uses an "inertial navigation system" to get a more fine-grained idea of where it is as it's moving.[6] This system uses sensors similar to speedometers and compasses embedded in the car. These sensors tell how fast the car is moving and in what direction.[7] The car can then estimate where it's gone after a certain amount of time has elapsed. Errors in these sensors can add up,[8] so the car might use map data to calibrate its position.[9]

Once the car knows where it is, it needs to understand what's going on around it, including other objects (both moving and stationary) and street directions (like lane markers or stop

signs).[10] This understanding begins with hyper-detailed maps of the area.

These aren't your garden-variety Google Maps, though; these maps are precise to the inch and include features like the height of every curb and the position of every traffic sign.[11] These maps are so important, in fact, that a war over high-precision map data has been brewing among carmakers: BMW, Daimler, and Audi spent $3.1 billion to acquire a 3D mapping company called HERE in 2015.[12] As you can imagine, Google's expertise with mapping and Street View has given it a huge head start in building hyper-detailed models of streets — a huge task, given that there are 4 million miles of road in the United States alone.[13]

The maps help the car understand core features of the roadway like curbs and lane markings.[14] The other difficult part is understanding the other objects on the road: pedestrians, cars, bikes, traffic cones, you name it.

Fortunately, self-driving cars come armed with a variety of sensors that help them out. A spinning laser mounted on the top of the car, called LIDAR, helps the car build a 360º model of its surroundings. Radar sensors help the car figure out how far it is from other objects.

These sensors help the car understand what objects are around it, but they can't help the car tell objects apart; a traffic cone and a parked car would "look" the same. To help with that, self-driving cars use cameras to identify objects based on their color and shape. Cameras also pitch in to help the car guess how far it is from obstacles.[15] The onboard computer helps the

car combine this information into a 3D model of its surroundings.[16] This model creates "boxes" representing pedestrians, cars, and bikes, and it also includes "fences" to show where the car might need to slow down or stop.[17]

A self-driving car prototype made by Google's affiliate Waymo. Notice the spinning LIDAR device and camera on top. Source: Wikimedia[18]

So far, the car knows where it is and has made a 3D model of all the objects around it. The car still needs to figure out the motion of these objects. To do that, the car has to predict where objects are going based on what it already knows.[19] For instance, it could predict that a car that has been decelerating for a few seconds is likely to keep slowing down.

But while the carmakers can encode some basic predictions, like "green means go," they can't possibly embed every single rule, so instead the car just learns based on what it sees.[20] For instance, self-driving cars could start noticing that, when bicyclists extend their left arm, they turn left 90% of the time.

So the next time the car sees a bicyclist to its right extending their left arm, the car could slow down and let the bicyclist pass. In other words, the car has just learned to avoid left-turning bicyclists![21] This technique is called machine learning; at its simplest, machine learning is just the art of making predictions based on observed patterns.[22]

The final step is for the car to make a driving strategy. Based on its current speed and position, the car makes a large set of possible actions, or "short-range plans," for how it could get closer to its destination: changing lanes, making a turn, accelerating, and so on. Then it removes plans that would bring it too close to an obstacle, and it ranks the remaining plans according to safety and speed. Once the car chooses the optimal plan, it sends those instructions to the wheels, brake, and "gas pedal" to make the car move. All of this computation happens within 50 milliseconds.[23]

The road ahead

So when will we be getting self-driving cars? They aren't quite ready for mainstream use yet, but self-driving car companies are busy testing out their cars. Test cars are on the road in Pittsburgh,[24] San Francisco,[25] and Austin.[26] The testing has been extensive: Google's self-driving affiliate Waymo said its cars had driven 3 million miles on city streets by 2016, and the cars logged a billion miles in simulations in that year alone.[27] Waymo even built an entire fake city in California to test its cars out.[28]

There are a wide range of projections for when self-driving cars will hit mass availability. Google said it wanted autonomous cars out by 2018,[29] and Nissan projected it would release self-

driving cars by 2020.[30] Other forecasters are more pessimistic, saying self-driving cars won't become mainstream till 2030.[31]

Don't forget that many cars already include some self-driving features. Tesla's Model S car unveiled a self-driving "Autopilot" mode in 2015.[32] Many other cars include features that rely on some self-driving technologies: self-parking, blind-spot detectors, collision avoidance systems, and drifting warnings, to name a few.[33] But totally self-driving cars, it seems, are a bit farther away.

Speed bumps

The technology behind self-driving cars is tough, but the regulatory issues are just as difficult. Self-driving car companies have been blocked by the government in a few situations. For instance, Uber tried to test its self-driving cars in San Francisco in 2016, but city regulators shut them down within a day for not having the right permit.[34] But other commentators complained that the government didn't have *enough* regulation on self-driving cars.[35] Fortunately for autonomous car fans, the government released its first rulebook on self-driving cars in 2016, finally setting some standardized laws that ended the regulatory "chaos."[36] These laws updated vehicle standards to no longer assume there was a human driver, clarified the difference between semi- and fully-autonomous vehicles, and set a "universal baseline technological and physical standard" that the cars must meet.[37]

Another issue is that of hacking. In 2015, a journalist found that hackers could remotely take control of his (not-autonomous) Jeep,[38] raising fears that hackers could do something similar to a self-driving car.[39] Even if self-driving

cars themselves are perfectly secure, clever hackers could find other ways to attack. For instance, one research group claimed that, by putting a small sticker on a stop sign, they could fool a self-driving car into not recognizing the sign and thus not stopping.[40]

Perhaps the hardest problem to solve, though, is one of computational ethics. The classic example is how cars should handle harm to humans. What should a self-driving car do when it has to choose between potentially injuring the driver or a pedestrian?[41] If self-driving car company programs its cars to make a decision, is the car programmed to kill?[42] To bring transparency to these ethical dilemmas, philosophers and technologists have called for "algorithmic transparency": the idea that self-driving cars' algorithms be made public.[43]

Despite these speed bumps, self-driving cars are likely to become more and more mainstream in the coming years. As we mentioned, self-driving cars will be great for consumer safety and convenience. But don't forget that there could be losers: in a world of self-driving cars, millions of people who drive for a livelihood could be out of a job.[44] That leads us to a bigger question: in a world of automation, who else's job will be at risk?

Are robots going to take our jobs?

Manufacturing robots have already put thousands of industrial workers out of a job, leading to an uptick in poverty.[45] A report issued in 2015 was even more grim: it predicted that automation would kill over 4.6 million office and administrative jobs by 2020.[46] In other words, both skilled and unskilled workers seem to be in trouble. It seems that robots will inevitably take our jobs.

Or will they?

The economics of tech and labor

Economists group technologies into two categories: labor-enabling and labor-replacing. Labor-enabling technologies help workers be more productive. For example, consider PCs and the internet — they've made it far easier to write essays, find information, or talk to coworkers. Then there are labor-replacing technologies, like the self-driving cars and industrial robots we mentioned before. As the name implies, labor-replacing technologies can remove the need for human workers. These opposing forces are in a constant tug-of-war.[47]

Who wins? The results might be unexpected. For instance, think about the ATM, which became popular in the 1970s. Most customers no longer needed to talk to a teller inside a bank's branch office. Many people assumed that this would eliminate the teller job altogether. Right? Wrong.[48]

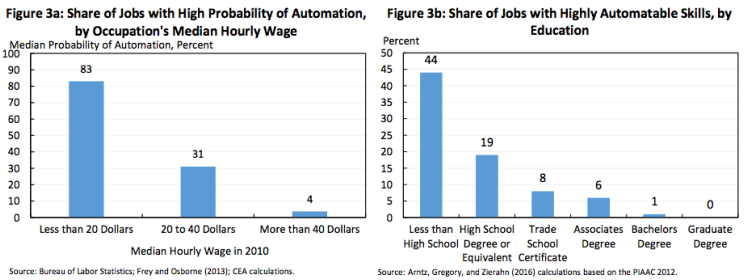
Thanks to the ATM, banks needed fewer human tellers in their branches. But this made branches cheaper to operate, which led to banks opening more branches. And that, in turn, led

banks to employ more tellers. The upshot? The number of tellers in the US grew from 300,000 to 600,000 from 1970 to 2010.[49] In other words, the ATM actually *created* teller jobs instead of replacing them.

So what does this mean for us in the age of artificial intelligence and robots?

The case for and against job theft

There's pretty strong evidence that automation will take away lots of jobs. A 2013 Oxford study found that half of all American jobs were at risk of automation by 2033.[50] People with lower skill levels would be particularly hurt. President Obama's Chief Economic Adviser found that 83% of jobs paying under $20 an hour were at risk of automation, versus just 4% of jobs making over $40 an hour.[51] Furthermore, 44% of jobs requiring less than a high school education were likely to be automated, compared to just 1% of jobs requiring a bachelor's degree.[52] In other words, robots could take our jobs, and it'd especially hurt the least educated and most vulnerable.



A 2016 report showed that people with lower-paying jobs, fewer skills, and less education were more at risk of losing their jobs to automation. Source: Council of Economic Advisors[53]

But there's also data to indicate that robots *aren't* taking our jobs. Throughout the mid-2010s, American unemployment was low (less than 5% in 2017), workers were staying at their jobs longer, and wages were slightly rising.[54] That hardly indicates that robots are massively stealing jobs. More broadly, automation could move people away from doing more manual tasks to doing more tasks that require the human mind. For instance, manufacturing plants envision having fewer assembly-line workers in the future — but more engineers, coders, and managers.[55] Technology has created entire industries, like IT and software development.[56] Automation wouldn't just create science, technology, engineering, and math (STEM) jobs, either: self-driving cars will still need mechanics and marketers, for instance.[57]

So what's the verdict? No one can agree. Pundits are split — sometimes hilariously so. For instance, one writer penned a New York Times piece saying, "The Long-Term Jobs Killer Is Not China. It's Automation."[58] But a Wired writer said, "[T]he answer is very clearly not automation, it's China." [59]

Haves and have-nots

We mentioned before that highly-skilled workers will have a lot to gain from automation, while low-skilled workers will have more to lose.[60] In other words, the poor would get poorer while the rich would get richer.

One way to mitigate this problem is education. There will be plenty of new jobs available but, unless something changes, there won't be enough skilled workers to fill them. In 2015, Deloitte estimated that manufacturing would add 3.5 million

jobs by 2025 thanks to automation — but 2 million of those would go unfilled due to a lack of skilled workers.[61] Proposals to combat this include apprenticeships to help people pick up skills on the job, investment in vocational education at community colleges, and promoting STEM education in high school and college.[62]

Some more radical suggestions have been proposed. Elon Musk projected that automation could send unemployment up to 30-40% (again, no one agrees on the impact of automation), so he proposed a "universal basic income." Under this scheme, governments would send every resident a check, which he said would fight poverty and protect the economy from certain collapse. Musk's universal basic income would be funded by a government tax on robots.[63]

As it happens, Bill Gates has long had a proposal to tax robots — or rather, to tax the companies that employ robots. Proponents say the revenue could fund jobs that humans are uniquely good at, like childcare.[64]

All this brings us back to our original question: are robots going to take our jobs? The evidence seems mixed, but it seems clear that robots will greatly help some people and greatly harm others. So what can we do about it? The best answer seems to be to invest more in people and make employees better trained to do the high-tech jobs of the future. Paradoxically, in an era where computers are taking over the world, our humanness might become more important than ever.

How does Siri work?

Ever since Apple launched Siri in 2011,[65] millions of people have come to rely on intelligent personal assistants like Siri, Google Assistant, Amazon's Alexa, or Microsoft's Cortana.[66] Whether you need to see today's weather, get driving directions, or send a text, intelligent personal assistants can help you solve a variety of problems — and sometimes even hold a conversation to boot.[67] But your phone doesn't have a brain. Instead, Siri has a form of "artificial intelligence"[68] — but what does this mean, and how does it work?

Natural language processing

To figure out how Siri works, let's step through an example. Say you want to see pictures of kittens, so you say *"cat"* to Siri. Making sense of your sounds takes a lot of computational power (as we'll see shortly), so to avoid overloading your phone, Siri sends an audio recording of your speech to Apple's powerful servers. (This explains why Siri doesn't work if you aren't connected to the internet.[69])

Once Apple's servers have your speech file, they try to turn your sounds into words. First, Apple breaks down what you said — which, phonetically, sounds like *"kat"* — into its individual sounds: *"k," "a,"* and *"t."* Then, Apple turns to a database that tracks how thousands of English speakers pronounce various words. Using this database, Apple would notice that the combination of the *"k," "a,"* and *"t"* sounds usually corresponds with the word "*cat*." And so, they'd infer that you meant "*cat*."[70] It might not seem like much, but Apple just turned an audio file into text!

Then, Siri tries to make sense of the words you said. If it notices words like "temperature" or "rain," Siri might tell your phone to open the weather app. If it notices a math question, it might open the mathematical search engine WolframAlpha. Otherwise, Siri could just run a web search.[71]

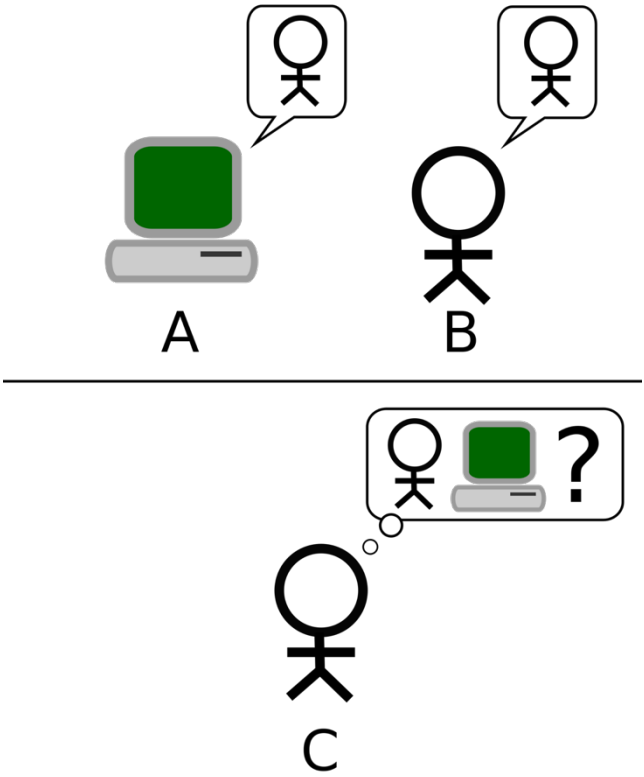
This whole procedure of turning speech into sounds, words, and ideas is called "natural language processing," or NLP.[72]

If Siri needs to say anything back to you, Apple converts its text to speech. This uses the same pronunciation database we mentioned earlier, just in reverse.[73] Finally, Apple's servers send the processed information back to your iPhone or Mac,[74] and Siri will speak.

Can Siri think?

In the 2013 film *Her*, a man falls in love with his intelligent personal assistant.[75] That raises the question: if apps like Siri can answer questions and hold conversations with us, does that mean they can think? Computer scientists have been debating this for generations, but we'll give a quick overview.

To answer the question of whether computers can think, the famous computer scientist Alan Turing put forward a thought experiment called the Turing Test in 1950. In this test, a person asks questions to either a human or a computer; the person isn't told who they're talking to. If the computer can trick the questioner into thinking that they're talking to another human, the computer passes the test.[76]



The Turing Test checks if a computer (A) can trick a person asking questions (C) into thinking they're talking to a human (B). Source: Wikimedia[77]

In 2016, a chatbot finally passed the Turing test, but technologists were generally skeptical, saying this didn't indicate that computers could actually think like humans.[78] Computer scientists classify most of the artificial intelligence we use today, like Siri and the chatbot we mentioned, as "weak artificial intelligence": something that's only good at specialized tasks.[79] Philosophers say that weak AI can only *simulate* how humans think instead of actually thinking.[80]

Meanwhile, artificial intelligence that could think as well as a human at a wide variety of tasks would be called "strong artificial intelligence."[81] When Stephen Hawking talked about doomsday scenarios like the "singularity," in which computers would evolve faster than humans and spell "the end of the human race," he was referring to strong AI.[82] But some philosophers think that strong AI will never happen.[83]

So, can Siri think? Not really. She can automate tasks, but truly thinking like a human seems far away — for now, at least.[84]

How could you make video and audio "fake news"?

In 2017, a Canadian company called Lyrebird published audio of Donald Trump, Barack Obama, and Hillary Clinton reading their Tweets aloud.[85] There was just one problem: none of them had actually read their Tweets — the audio was faked![86]

Thanks to a new technology, you can create convincing but fake video and audio in a fraction of the time it would take you to manually imitate it.[87] In this era of fake news, many people have stopped believing news they hear online, relying on video or audio to confirm that something really happened. But if video and audio can also be faked, we wouldn't have anything left to trust.[88] So how can you make fake video and audio footage?

This mysterious technology is called "generative adversarial networks," or GANs. GANs are a special version of a technology called neural networks.[89] So before we explain how GANs work, let's look at neural networks.

Neural networks

Your brain learns things by experimenting, getting feedback, and adjusting.[90] For instance, say you're new to baking and want to bake a cake. You throw together a random mix of flour, sugar, eggs, butter, and other ingredients, toss the batter in the oven for a random amount of time, and see how it turns out. A friend tastes the cake and gives you comments: too sweet, undercooked, needs more chocolate. You slightly adjust your recipe to address the comments, and while your friend

might be happier, they still have suggestions. Repeat this process, and eventually you'll have tweaked your approach enough to become a master cake baker — without ever picking up a cookbook. You learn this way because your brain contains a "neural network":[91] a collection of connected cells called neurons that talk to each other.[92]

To make computers more powerful, computer scientists have tried to simulate your brain's neural network inside a computer. It's called an "artificial neural network," but many technologists (somewhat confusingly) just call it a "neural network."[93] Like the cake example, an artificial neural network keeps track of many variables and assigns each variable a "weight": the amount of butter, the time to bake, or the oven temperature, to name a few.[94] As you give the artificial neural network feedback, it tweaks the weights to get closer to the right answer, much like you adjusted your cake recipe based on your friend's feedback.[95]

Neural networks are incredibly powerful: they can autocorrect text on your phone, catch spam emails, translate languages, or read your handwriting, among many other things.[96] Neural networks are great at recognizing things, but they aren't designed to generate new stuff, like fake audio and video.[97] For that, let's turn to an even stronger variation of neural networks.

Generative adversarial networks

In a "generative adversarial network," or GAN, you create two neural networks and make them face off. The "generator" network tries to make something fake, and the "discriminator" guesses whether or not the generator's creation is real or not.[98] The networks get into a sort of arms race, with the generator

trying to make more convincing forgeries and the discriminator trying to get better at policing fakes.[99] The networks learn from each other, constantly improving, until the generator is churning out incredibly convincing fake things.[100]

As an example, imagine you want to make a generative adversarial network that creates a fake video of an American CEO giving a speech. You create the generator and discriminator network. Originally, neither network has any idea what it's doing, so the generator might make a video of a person speaking Italian, and the discriminator won't know that it's fake. So you step in and show the discriminator a real video of an American CEO speaking. From the video, the discriminator learns that American CEOs usually speak English. So the discriminator starts rejecting videos of people speaking other languages. The generator picks up on this, so it starts trying different languages to trick the discriminator. Eventually the generator discovers that videos of a person speaking English pass the discriminator. This back-and-forth continues until the generator can make convincing fake videos.[101]

So, what would we do if we started seeing faked videos of presidential candidates giving inflammatory speeches?[102] No clue — but at least we'd know how the fakers made the videos.

Could Amazon be the first trillion-dollar company?

In September 2017, Apple was the world's most valuable company, with a market capitalization (or total value of all shares) of $834 billion. Amazon was fifth at $470 billion.[103] But in 2016 and 2017, a flock of analysts predicted that Amazon would soon become the world's first $1 trillion company. One analyst at Barclays said it was a virtual certainty.[104] An investment bank director named Mark Mahaney thought Amazon would hit $1 trillion 2021,[105] and NYU professor Scott Galloway was even more confident, predicting that Amazon would reach the milestone in 2020.[106]

The investors were certainly on to something, since Amazon's stock price almost quadrupled between 2013 and 2017.[107] But as of 2017, Amazon wasn't even halfway to $1 trillion — so what made investors so optimistic? And what does this prediction mean for the future of Amazon and the tech industry?

Amazon's stock price has been growing explosively since 2013. Graph made in September 2017. Source: YCharts[108]

Sector-by-sector dominance

Much of Amazon's dominance comes from its strength in a huge range of industries. Besides retail, Amazon is involved in hardware, media, gaming, book publishing, clothing, cloud computing, and groceries.[109] And that's not even all of it. But to explain how Amazon is well positioned to become the first $1 trillion company, we'll explore Amazon's remarkable power across several industries.

We'll start with retail, Amazon's bread and butter. Amazon pulled in $107 billion in online retail sales in 2015, dwarfing Walmart's $13.7 billion in online sales in the same timeframe.[110] Amazon is the world's largest online retailer, hands down.[111] And there's every reason to believe that this domination will continue to grow. For one, Amazon is eating brick-and-mortar retailers for breakfast[112] at a time when these stores are shuttering at record paces: 8,000 retail stores closed in 2017, compared to 4,000 in recession-stricken 2008.[113] Since Amazon's core business doesn't rely on brick-and-mortar stores, it can grow as fast as it wants for free; meanwhile, even mighty Walmart needs to spend $37 million to open a new store.[114] Amazon's massive scale and low costs help it undercut any retailer it wants, driving many out of business.[115]

Amazon is taking a larger and larger slice of the online retail market too.[116] The first reason is its delivery infrastructure, which is second to none.[117] Acquiring Whole Foods's 400 stores gave Amazon 400 more places to ship food from, which boosted Amazon's delivery infrastructure even further.[118]

Moreover, Amazon is becoming synonymous with shopping for many Americans,[119] to the point where even Google and Facebook have reason to be worried. In 2016, 55% of American shoppers began their online search at Amazon, up from 44% a year before; search engines' share dropped from 34% to 28%.[120] In short, Amazon dominates online and offline retail, and evidence suggests this trend will continue.

Amazon's other star sector is the cloud.[121] In 2017, Amazon Web Services dominated: it had 34% market share, while its biggest rival — Microsoft's Azure — was a distant third with just 11%.[122] And AWS is a huge asset for Amazon: its revenue soared 55% in 2016,[123] and it brings the majority of Amazon's profits.[124] In other words, if Amazon ever needs cash, AWS will provide plenty of it.

With Amazon's purchase of Whole Foods, they entered the $700 billion grocery market.[125] According to Scott Galloway, this sector was "ripe for disruption."[126] Amazon entered the grocery world with a splash, chopping prices at Whole Foods by up to 43% as soon as it bought the chain.[127] Thanks to its endless cash supply and strong logistics, many experts believe that Amazon will take over the grocery business the same way it took over retail.[128]

The last sector we'll cover, and a bit of a dark horse, is business-to-business (or "B2B") sales. B2B is a $8 trillion market,[129] and one research group predicted that online B2B sales would be twice as big as Amazon's normal business-to-consumer, or B2C, sales.[130] Amazon's 2015 launch of Amazon Business, a B2B platform selling everything from bulk food to

janitorial supplies, shows that the company is thinking about entering the market.[131]

It looks like the Everything Store is set for fabulous growth. But it might not end happily ever after.

Antitrust action?

After Amazon bought Whole Foods in 2017, many commenters started asking if Amazon had finally become monopolistic, and if so, what the government should do about it.[132] Do they have a strong argument?

First, let's review the laundry list of complaints about Amazon's allegedly anti-competitive behavior. Amazon has become notorious for fiercely undercutting competitors, driving them out of business and preventing new competitors from entering the market.[133] With the Whole Foods Purchase, Amazon has been attacked for blocking any future grocery delivery startups from entering the market at all.[134] And even when new retailers gain a foothold, Amazon tries to purchase them. For instance, in the 2000s Amazon snapped up Diapers.com and the shoe store Zappos.com, both of which were the biggest online retailers in their respective categories.[135]

However, American antitrust laws seem ill-prepared to take on Amazon, primarily because these laws are designed to prevent head-to-head competitors from merging, like when the US blocked the ISP giants Comcast and Time Warner Cable from merging in 2015.[136] In contrast, Whole Foods was in an entirely different market than Amazon. Regulators could take the approach of saying that, through a series of acquisitions, Amazon has acquired so much market power that it can force

suppliers to sell at too-low prices and unfairly undercuts its competition.[137]

So, will an antitrust case be brought against Amazon, and would it do anything to the tech giant? Will Amazon coast to a trillion dollars, or will it face stiff competition? We can't say, but we're sure some interesting battles are going to play out in the coming years.

Conclusion

That's a wrap for *Swipe to Unlock*. Give yourself a pat on the back: you've learned an incredible amount about technology and business strategy behind it. The next time you hear an entrepreneur talk about their "cloud-based SaaS app" or rave about their "machine learning big data platform," you'll be able to understand them — and maybe even strike up a conversation. When you hear hot topics like "ransomware" or "net neutrality" on the news, you'll be able to think critically and cut through the buzzwords. And if you're ever working with engineers or businesspeople in the tech industry, you'll be able to ask the important questions: *What APIs are you using? What's the monetization strategy? What have your A/B tests found?*

Reference material

Now that you've learned the main content of *Swipe to Unlock,* we hope this book will become a useful handbook for you going forward.

In the glossary, we've defined dozens of important technology and business terms — including some of the ones we talked about above. The glossary even contains some information we didn't get time to cover, like explanations of some popular programming languages and business terms.

We've also included an index where you can look up particular companies, products, concepts, and people. If you're interviewing at a tech company, for instance, you could look up the company and read the key sections about them and their products.

Free bonus chapter

But before you go, we have one more tidbit up our sleeves.

These days, it seems like you can't turn on the news without hearing about Bitcoin, blockchain, or cryptocurrencies. But what are they? How do they work? And are they for real, or just a fad? Get the lowdown in our free bonus chapter, available at swipetounlock.com/bonuschapter.

Thank you!

We hope you've enjoyed *Swipe to Unlock* and that what you've learned here is useful for your life and career. Writing this book has been a ton of fun for us, so we're glad you joined us for the ride. If you found *Swipe to Unlock* useful, we hope you'll recommend it to your friends or write a review on Amazon.

Thank you for reading, and until next time, all the best!

— Neel, Parth, and Adi

Glossary

So how do you build a website? Just ask a technology help website, and it'll tell you: open a GitHub repo; build out a backend in Python or Ruby on Rails; serve up some HTML, CSS, and JavaScript; make some RESTful APIs; tweak the UI/UX; and ship an MVP to AWS. Oh, and get a CDN while you're at it.

Huh?

The world of software has a crazy amount of jargon and buzzwords. But here, we're going to break down some of the most common terms — including the ones we just mentioned — so you can talk like a techie without feeling like you need a foreign language class.

Programming languages

All software is written using code, much like food is made according to a recipe. And just like you can write recipes in English, Bengali, or Turkish, you can write software in programming languages like Ruby, Python, or C. Each programming language has its own strengths and weaknesses, and each one is used in particular situations. Here's a glimpse into some of the most popular languages.

Assembly

Computers only think in 1's and 0's, and assembly language is just a slightly prettier version of 1's and 0's. Programmers rarely write in assembly language, since it's too much effort; they'll usually write in a "higher-level language" that computers convert into assembly and then run. (Every other language in this section is a higher-level, or more "abstract," language.) It's like driving a car: instead of trying to directly set the speed of each wheel, you just use the steering wheel and pedals. This is far easier, and besides, you'd probably have no idea how to set the wheel speeds just right.

C/C++

Some of the oldest programming languages, and still among the most popular. They run really fast but are more difficult to write, so developers trying to get maximum performance (like those writing graphics-heavy games, physics simulators, web servers, or operating systems) often use C and C++.

C# (C-Sharp)

A language built by Microsoft, often used to write desktop apps. Similar to Java.

CSS

A web development language that works with HTML, used to make websites look pretty. CSS lets you change a webpage's colors, fonts, background, and so on. It also lets you specify where the various buttons, headers, images, etc. on a webpage should go.

Go

A language built by Google, often used to write web servers.

HTML

The language used to write webpages. You can create links, images, headers, buttons, and every other element on a webpage using HTML. Each of these elements is called a "tag." For instance, a tag called <img> represents an image.

Java

One of the world's most popular languages, Java is used to write Android apps, web servers, and desktop apps.

JavaScript

The language used to make webpages interactive. Every web app you use, from Facebook Messenger to Spotify to Google Maps, uses JavaScript. Nowadays, developers also use JavaScript to build web servers and desktop apps.

PHP

A language used to write web servers. Facebook is written using a custom "dialect" of PHP.[1]

Python

A popular, easy-to-learn language that's common in introductory CS courses. It's widely used for data science and writing web servers.

R

A data analysis language, which lets you graph, summarize, and interpret huge amounts of data.

Ruby

A language commonly used for building web apps through its popular web-server software Ruby on Rails.

SQL

Structured Query Language: a language for working with databases. Like Excel, it lets you work with tables, rows, and columns, as well as specifying far more complex analyses, called "queries."

Swift

Apple's language used for writing iPhone, iPad, and often Mac apps.

Data

We humans like storing information in Excel files or Word documents. Computers, however, prefer storing data in simple text files. Here are a few popular ways to store data in a "machine-readable" format.

CSV

Comma-separated values: a way to store data in lightweight tables, similar to an Excel file but much simpler. These files have the ".csv" ending.

JSON

A popular data storage format, often used by web apps. It's more free-form than CSV, allowing data objects that are nested inside *other* objects. For instance, a "person" object could contain "name" and "age" data.

XML

Another text-based data storage format. Like HTML, it stores and organizes data using tags, and like JSON, it allows nesting.

Software development

To talk like a software developer, you need to know these common terms and buzzwords. Let's break them down.

A/B tests

Running experiments to decide which features to put into a product, usually a web-based one. You show some users one variation of a feature, and other users a different variation. For instance, Amazon could show half its users a red "buy now" button and the other half a blue button. Then they'd look at various metrics, like number of sales or number of clicks, to decide which variation is better, and they'd roll out the winning variation to all users. Product managers and developers love A/B testing, since it helps them scientifically determine how to improve their software.

Agile

A software development paradigm that emphasizes short, alternating bursts of writing software and getting feedback from users. For example, instead of taking months or years to release one huge final product, an Agile team would prioritize quickly releasing a "minimum viable product," or a simple prototype. Then the team would get feedback from users to "iterate on" and improve the prototype, repeating this process many times until it is happy with the result.

Angular

Google's web development framework, often used to build web apps. Several popular websites, like Tesla, Nasdaq, and The Weather Channel, use Angular.[2]

Backend

The "behind-the-scenes" part of an app or website that users don't see. The backend handles and stores data. For websites, the backend prepares the webpages that are eventually shown to the user. An analogy: in a restaurant, the cooks in the kitchen are the "backend", since they prepare the food that customers enjoy, even if the customers never see them.

Beta

A preliminary version of software, often released to testers to get user feedback before the final product launches.

Big data

Working with huge amounts of data to extract interesting insights. There's no specific definition of how big counts as "big", but if a dataset is too huge to fit on a single normal-sized computer, there's a good chance that it's "big."[3]

Blockchain

The technology behind Bitcoin, blockchain allows for decentralized transactions. Imagine you could hail an Uber without having to use the Uber app or send someone a message without a company like Facebook or your cell phone provider getting in between. With blockchain, everyone shares a record of every past interaction, so you don't need a central authority. With Bitcoin, every user has a list of every past sale, so no one person or company "owns" Bitcoin. This also protects against fraud, since everyone knows if one person is trying to pull something shady.

Bootstrap

A popular toolkit for designing websites, Bootstrap is basically a giant CSS file that contains nicely-designed

layouts, fonts, colors, and other styling options. Many websites use Bootstrap as a starting point for their styling; it's a very powerful website template.

Caching

Storing information in a particular place on a computer so you can access it more quickly. It's like storing your favorite pizza place's phone number in your contacts so you don't have to Google it every time — it makes recalling the information faster.

Cookie

Small notes that websites store on your browser to remember your customizations. For instance, an e-commerce site could store your cart, preferred language, or username in cookies. Cookies also enable targeted advertising: websites can pass around your location and other personal information via cookies to figure out what you like and hence what ads to show you.

Database

A giant table used to store information; like a superpowered Excel file. For instance, Facebook might store information about all its users in a database, with a separate row for each user and columns for name, birthday, hometown, etc.

Docker

A way for developers to package up everything an app needs to run in a "container," which anyone can run on any supported machine. It's convenient because you don't have to worry about having the right system configuration; the same container will run the exact same way everywhere. Docker is much more efficient than the alternative, which is to boot up a whole new operating system to run each app.

Flat design

A minimalistic design trend, where you remove unnecessary shiny colors, shadows, animations, and other details, reducing the app to simple colors, geometric shapes, and grids. A few examples are Microsoft's Metro UI (the tiled design used in Windows 8 and 10)[4] and Apple's flattening of iOS in version 7.[5]

Frontend

The user-facing part of a website or app. The frontend includes all the buttons, pages, and pictures that users interact with. It takes information from the user, sends it to the backend, and updates what the user sees once the backend responds. As an analogy, the waiters in a restaurant are the "frontend". Waiters take diners' requests to the cooks (the backend) and serve diners the completed food.

GitHub

A website that hosts millions of open-source software projects. Anyone can see and build on others' code here. Code on GitHub organized into repositories, or "repos." People can "fork" these repos to make their own versions of the code, and developers can suggest changes to repos using "pull requests."

Hackathon

A coding competition where developers team up to build cool, creative software in short sprints, often from 12 to 72 hours. Hackathons often feature prizes, tech company recruiters, free swag like t-shirts and stickers, and late-night food.

Hadoop

A free "big data" software package for storing and analyzing huge amounts of data — we're talking terabytes and petabytes.

jQuery

One of the most famous web development libraries, it's a JavaScript tool that makes interactive websites much easier to build.

Library

A reusable chunk of code that one developer publishes online for other developers to use. For instance, D3 is a famous library that lets JavaScript developers make interactive graphs, charts, and maps with just a few lines of code. *Also known as package or module.*

Linux

A free, open-source family of operating systems; alternatives to Windows and macOS. Many of the world's biggest supercomputers, as well as most web servers, run Linux. Android is built on Linux, too.

Material design

Google's design theme, used for Android and many Google apps. It features bright colors, square "cards" of information, and sliding animations. It's similar to flat design, but it has some shadows, gradients, and 3D elements that flat design wouldn't have.[6]

Minification

A technique that developers use to compress their code files by removing any unnecessary bits of text. *Also known as uglification or compression.*

Mockup

After wireframing and prototyping, designers make mockups, which are high-quality drawings that specify

exactly which fonts, colors, pictures, spacing, etc. the developers making the app should use. Mockups help designers ensure every detail is perfect and get feedback before the app is actually coded. As UXPin says, "Wireframes are the skeleton. Prototypes demonstrate the behavior. Mockups are the skin."[7]

Node.js

A JavaScript framework for building web apps.

Open-source

A philosophy for building software where anyone can see, copy, and improve the code behind an app. (It's as if a restaurant let you see the recipes behind its dishes.) Many popular apps and platforms are open source, like Linux, Android, Firefox, and WordPress.[8] Many programming languages and software development tools are also open-source.[9]

Persona

Example people that designers create to summarize the types of users in their market. Personas have names, backstories, and personalities.[10] For example, LinkedIn's personas could be Sanjana the Student, Ricky the Recruiter, or Jackie the Job-Hunter.

Prototype

An early version of an app or website that lets app makers test their ideas with users. Prototypes can be as complex as clickable websites or as simple as stacks of sticky notes.

React

Facebook's web development framework, often used to build web apps. Websites like Facebook, Instagram, Spotify, The New York Times, Twitter Mobile, and many others use React.[11]

Responsive web design

Making websites work well on all screen sizes — phones, tablets, laptops, etc. For instance, the New York Times might show several columns of text on bigger screens (and the print paper), but just one column on smaller screens.

Ruby on Rails

A framework for building web apps using Ruby. Airbnb, Twitch, and Square are all built with Ruby on Rails.[12] *Also known as RoR or Rails.*

Scrum

An offshoot of Agile, where software development teams release new features every few weeks, organizing their work into "sprints." They often have daily 15-minute "stand-up" meetings so everyone knows what everyone else is doing and so that vital information is shared across the whole team.

Server

Computers that power websites and many apps. Servers tend not to have a graphical interface, screen, mouse, microphone, or other gadgets. Instead, they're used solely for their computational might.

Stack

The suite of technologies that an app or website is built with. This includes the app's choice of frontend tools, backend tools, and database. As an analogy, a car's "stack" could include a particular kind of upholstery, engine, tires, and headlights, among other things.

Terminal

A text-based interface available on computers that developers use to build software. Even if you're not writing code, the terminal is handy for complex customization, and a few apps can only be run from the terminal, not from the

point-and-click interface we're used to. *Also known as command line, shell, or Bash.*

Unix

A family of operating systems, including Linux and macOS.

Wireframe

A simple way to draw the "skeleton" of an app or website,[13] like how you might make an outline before writing an essay. Wireframes are made of just lines on paper: buttons and images become boxes, sidebars become rectangles, text becomes squiggly lines, and so on. Wireframes help figure out where page elements should go before anything gets coded.[14]

Technology's alphabet soup

Acronyms might just be the most frustrating part of software jargon. Here are a couple of the most common ones.

AJAX

A method for one website to access information from another using an API. Uses JavaScript.

API

Application Programming Interface: A way for one app to get information from another app, or make another app do something. For instance, Twitter has an API that lets another app post Tweets on someone's behalf, and ESPN has an API that lets you grab the latest sports scores.

AWS

Amazon Web Services: a platform that lets you store data or run apps in the cloud.

CDN

Content Delivery Network: a way for websites to serve images, CSS files, and other "static" assets faster by using a separate dedicated website. These dedicated CDN websites are specialized for holding files instead of running code, and they have many servers scattered around the world, so anyone can get the files much faster than normal.

CPU

Central Processing Unit: the "brain" of a computer or phone, which runs the operating system and apps.

FTP

A protocol for sending files to and from web servers.

GPU

Graphics Processing Unit: a special part of a computer optimized for drawing graphics. If you ever hear the term "hardware-accelerated animation," that uses the GPU.

HTTP

HyperText Transfer Protocol: a protocol used to view webpages on the internet. By "protocol," we mean a set of rules for how information should be transferred.

HTTPS

HyperText Transfer Protocol Secure: an encrypted version of HTTP, which is used for secure online communications like banking, payments, email, and logging into websites.

IaaS

Infrastructure-as-a-Service: tools that let you rent out another company's server space to run your app. One example is Amazon Web Services.[15]

IDE

Integrated Development Environment: a specialized app that makes it easy for developers to build particular kinds of software. Eclipse, for instance, is an IDE for Java and

Android. It's like how chefs have their own specialized kitchens with particular tools and ingredients.

I/O

Input/Output: the process of reading and writing files.

IP

Internet Protocol: a protocol for moving "packets" of information from one computer to another over the internet. Works closely with TCP. HTTP is built on top of TCP and IP.[16]

MVC

Model-View-Controller: A way of organizing code, which often builds on Object-Oriented Programming. Many web or app development frameworks use MVC.

MVP

Minimum Viable Product: in Agile, an early-stage prototype used for early testing. For instance, consider the MVP of the online shoe seller Zappos. The founders took photos of shoes at local stores and posted them on a website — and whenever someone "purchased" a shoe, the founders would buy the shoe from the store and mail it to them.[17] MVPs are just a simple version to see if people like the idea.

NLP

Natural Language Processing: a form of artificial intelligence that deals with understanding human languages.

NoSQL

A way of constructing databases, an alternative to (you guessed it) SQL. NoSQL emphasizes more free-form interaction with data rather than just working with rows and columns, which SQL does.

OOP

Object Oriented Programming: a way of structuring code so it's easier to understand, reuse, and build on. You represent everything as an object, from interface elements like *Button* or *Picture* to concepts like *Customer* or *Dog*. For instance, Snapchat could have objects like *User*, *Snap*, *Group*, *Sticker*, *Story*, or *CameraButton*. Each object has its own associated information and actions; for instance, a *Dog* could know its name and know how to bark.

PaaS

Platform-as-a-Service: tools that run an app for you; you just need to send them your code.[18] Between IaaS and SaaS in terms of complexity.

RAM

Random-Access Memory: a computer's "short-term" memory, which apps use to store temporary information like which browser tabs you have open. The more RAM your device has, generally, the faster it is.

REST

A popular type of API. APIs of this type are called RESTful.

ROM

Read-Only Memory: information that's burned onto hardware and usually can't be changed. Computers store the code needed to start the computer in ROM. *Also known as firmware.*

SaaS

Software-as-a-Service: software that's delivered over the internet, meaning you'll often use it in your web browser. Google Docs is a classic example. You'll often need to pay for SaaS apps on a monthly or yearly basis instead of paying to download the app upfront.

SDK

Software Development Kit: a pack of tools that help developers build apps for a particular platform, such as Android or Google Maps.

SEO

Search Engine Optimization: changing your website so it shows up higher in Google search results. One example is including the right keywords in your page's title or headings.

SHA

A popular cryptography algorithm used for encoding and decoding secure communications. There are multiple versions of SHA; as of 2017, the most modern one is SHA-3.[19]

TCP

Transmission Control Protocol: a protocol for breaking information into smaller chunks, so you can send it over the internet more easily.

TLD

A domain name ending like .com, .org, or .gov. Each country has its own TLD, called a "ccTLD": France has .fr, Mexico has .mx, India has .in, and so on.

TLS

Transport Layer Security: a method of encrypting information sent over the internet so hackers can't snoop on communications.

UI

User Interface: a type of design focusing on making apps and websites look good. Deals with colors, fonts, layout, etc. Often paired with UX.

URL

Uniform Resource Locator: A webpage's address, such as "https://maps.google.com" or "https://en.wikipedia.org/wiki/Llama".

UX

User Experience: a type of design focusing on making apps and websites easy to use. Deals with how to arrange parts of a website and webpage. Often paired with UI.

Business side

Not to be outdone by the software developers, tech companies' businesspeople — think marketers and strategists — have their own favorite jargon.

B2B

Business-to-business: companies that normally sell to other businesses instead of to average people like you or us. Some famous B2B tech companies are IBM, which sells cloud computing services to businesses, and Accenture, which provides technical consulting.[20]

B2C

Business-to-consumer: companies that sell to consumers; in other words, you could buy their stuff from stores or websites. For instance, Fitbit, Nike, and Ford are B2C. Some companies could be both B2B and B2C. For instance, Coca-Cola sells soda to shoppers but also to universities, hotels, and restaurants.[21] And Microsoft sells Office to both consumers and large businesses.

Bounce rate

How often visitors to your app or website leave without doing anything meaningful, such as clicking a link. A high bounce rate might suggest that visitors aren't interested in what the website has to offer.

Call-to-action (CTA)

A button or link that prompts visitors to take some action, like "Join our mailing list" or "Register for our free conference."[22]

Churn rate

The percent of users that a company loses over a particular timespan. For instance, if 1,000 people signed up for

Office 365 but only 750 renewed their subscription the next year, the churn rate would be 25%.

Cost-Per-Click (CPC)

A common type of internet ad, such as the ones seen on Google, that charge advertisers a small fee every time someone clicks on their ads. Also known as Pay-Per-Click (PPC).

Cost-Per-Mille (CPM)

A type of internet ad. Advertisers pay a flat fee each time 1000 people view the ad on a website, such as in Google search results. Also known as Pay-Per-Impression (PPI).

Click-through rate (CTR)

The number of people who clicked on an ad divided by the number of people who saw it and had the option to click on it. In other words, this is the likelihood that an average person would click on the ad. It's a way to measure how successful an ad was.

Conversion

Whenever a user does something that the business wants; the precise action can depend on the company's goals. Conversions could include joining the mailing list, signing up for an account, or buying an item.

Customer Relationship Management (CRM)

Software that a company uses to track its relationships with customers and business partners. Companies can track emails, meeting notes, and other data with CRM software.[23]

Funnel

A metaphor for how the pool of potential customers shrinks before they make a particular "conversion," like buying a product. For instance, an e-commerce website might get 1,000 visitors, but only 500 would search for

something, 100 would put something in their cart, and 50 would make a purchase.

Key Performance Indicator (KPI)

A metric that companies use for tracking success of products, teams, or employees. For instance, YouTube's KPIs could include number of users, number of videos, or number of watches.

Landing page

A small webpage targeted at a particular demographic; it'll often offer visitors something useful like an eBook or mailing list in exchange for their contact information. In other words, it's a targeted way to acquire leads.[24]

Lead

Someone who's shown interest in using a service or buying a product. Marketers try to turn strangers into leads and leads into customers, a process called "inbound marketing."[25]

Lifetime value (LTV)

How much money a customer will bring you, directly or indirectly, over the duration of your relationship with them. For instance, if a college bookstore thinks that students will spend $500 a year on textbooks over 4 years of school, each student's lifetime value would be $2,000. Generally, companies will only try to acquire a customer if their lifetime value is higher than the cost of acquiring them.[26]

Market penetration

How much of a target market a product or industry actually reaches. For instance, there are about 30 million teenagers in the US,[27] and if a teen-focused social network had 6 million teenaged users, it'd have 20% penetration of the teenager market.

Market segmentation

Breaking down a huge, diverse market into smaller, more specific ones. For instance, a company could segment its market by gender, location, interests (also known as "psychographics"), and income (part of so-called "behavioral" segmentation).[28]

Net Promoter Score (NPS)

A metric that measures customer satisfaction. Customers are asked to rate a product or service on a scale from 0 (they hate it) to 10 (they love it).[29]

Return on Investment (ROI)

The ratio of a project's profit to its cost.[30] For instance, if you spent $2,000 on an ad campaign and sold $2,600 of extra software, your ROI would be 30%. ROI is just a synonym for "bang for your buck."

Small- and medium-sized businesses (SMBs)

Generally, businesses with under 1,000 employees.[31]

Value proposition

A short statement that explains why consumers would find it useful. For instance, in 2015 the e-book website Scribd used the value proposition "Read like you own every book in the world."[32]

Year-over-year (YoY)

The change in a metric between a given point and a year earlier. This is useful when there are seasonal variations in the metric. For instance, if educational software sales are always low in summer, it doesn't make sense to compare this June's sales to this March's. Instead, you'd compare to last June's.

Roles at tech companies

Tech companies hire "standard" kinds of professionals, like marketers, CEOs, and HR representatives. But software is made differently than most physical goods, so tech companies have special kinds of roles. Here's a quick glimpse inside the software industry.

Backend engineers

Software engineers that handle databases and web servers. For instance, Facebook's backend engineers write the code that lets Facebook's supercomputers store billions of photos and handle billions of daily visitors. *See also software engineer.*

Data scientist

Data scientists analyze the company's data (on customers, sales, usage, etc.) to inform the company's business strategy and products.

Designer

Designers make apps and websites beautiful and functional, and they also design stuff like logos, colors, and branding.

Frontend engineers

Software engineers who build the customer-facing apps and websites. For instance, Facebook's frontend engineers make the Facebook website and apps look and work well. *See also software engineer.*

Product Manager (PM)

PMs sit at the intersection of business, design, and engineers. Based on what the customers and business need, they decide what products (apps, features, or even hardware) to make and what features the products need to

have, then work with engineers to make the products happen. Think of them as the conductors of the orchestra: they help all the various parts work together to make a great piece of music (or software, in this case).

Product Marketing Manager (PMM)

A slightly more marketing-focused version of Product Managers, they're more focused on launching and marketing products instead of building them.

Quality Assurance engineers (QA)

These engineers rigorously test software and hardware to hunt down bugs and ensure the software is robust.

Software engineer

People who write code and build out software. *Also known as SWE, software developer, or dev.*

Acknowledgements

Very soon after we started writing *Swipe to Unlock*, we realized that we couldn't do it alone — it takes a village, as they say. Here, we'd like to thank all our friends and family who provided their support, feedback, and inspiration as we were writing this book.

Neel

Thanks a million to Alaisha Sharma, Amy Zhao, Andrea Chen, Aron Szanto, Arpan Sarkar, Ivraj Seerha, Jeffrey He, Maitreyee Joshi, Menaka Narayanan, Saim Raza, Sathvik Sudireddy, Sohum Pawar, Tara Mehta, and Vishal Jain for your suggestions and advice on *Swipe to Unlock*. All my other friends deserve a shout-out for putting up with me talking nonstop about this book. I'd also like to thank some of my mentors who have inspired me to use my tech skills to make a difference: Jeff Meisel, Professor Latanya Sweeney, Nick Sinai, Seamus Kraft, and William Greenlaw, to name just a few. My fantastic co-authors, Adi and Parth, have been some of the best friends and colleagues. And finally, I'd like to thank my parents for their endless support through whatever I do.

Parth

A huge thank you is in order for all the friends and family who helped us get this book to where it is today. I'd like to especially thank Michelle Wang, Deborah Streeter, Jeremy Schifeling, Jack Keeley, Christina Gee, Stephanie Xu, Niketan Patel, Kevin Cole, Bradley Miles, Krishna Detroja, Gabrielle Ennis, Adam Harrison, Donna Haeger, Amanda Xu, William Stern, and Samantha Haveson. These kind individuals provided their valuable time, skills, opinions, and insights on everything from content to design to marketing. I am incredibly grateful for having the support of such amazing friends and family.

Adi

I would like to thank everyone who patiently helped us along the way — we wouldn't have been able to do it without your support. First, a sincere thank you to Pam Silverstein, Peter Cortle, Professor Nancy Chau, and Professor Michael Roach who enthusiastically encouraged us to share our technology knowledge. Next, I'd like to thank all my friends who helped with everything from reviewing chapters to looking over cover designs — thanks Michelle Jang, Lauren Stechschulte, Holly Deng, Eunu Song, Sai Naidu, Sandeep Gupta, Nivi Obla, Jenny Kim, Brian Gross, Eric Johnson, and Eileen Dai. Also, a huge shout-out to Natsuko Suzuki who tirelessly helped out with the design heavy lifting for the book cover, website, and other branding. And finally, thanks to my family for their support and love throughout this journey.

Index

If you want to learn more about a particular company, app, technology, or concept, here's where you'll find the appropriate pages.

Note that page numbers next to a specific product, such as Amazon Kindle, indicate sections about those products. Page numbers next to a company name indicate sections that are about the company in general, like how Amazon makes money.

Notes

There's so much to know about technology and the business strategy behind it, and in Swipe to Unlock we've only scratched the surface. That's why we're providing links to every single source we used during our research. To keep the paperback slim, we've put the links on our website at http://swipetounlock.com/notes/2.1.0/. If a particular fact or opinion piques your interest, we encourage you to read the source and dive deeper into the material!