Practical Recommender Systems

Kim Falk

MANNING

about the author

KIM FALK is a data scientist who is experienced building data-driven applications. He's passionate about recommender systems and machine learning in general. He has trained recommender systems to provide movie choices to end users as well as ads to people, and has even helped attorneys find case law content. He's worked with Big Data solutions and machine learning since 2010. Kim often speaks and writes about recommender systems. You can find him at http://kimfalk.org.

When he isn't teaching machines to stalk people, Kim is a family man, father, and trail runner with his German Pointer.

1 What is a recommender?

It's a jungle out there as far as understanding what a recommender system is, so we'll start this book looking into what problems it solves and how it's used. Here's what we'll cover:

- Understanding the task a recommender system is trying to emulate
- Developing insight into what are nonpersonalized and personalized recommendations
- Developing a taxonomy of how to describe recommenders
- Introducing the example website MovieGEEKs

Get a cup of coffee and a blanket and make yourself comfortable for this introduction to the world of recommendations. We'll ease into it, first looking at real-world examples before moving into the computational intricacies of a recommender system in the following chapters. You might feel tempted to skip ahead, but don't. You need the basics to understand what the result of your recommender engineering efforts should be.

1.1 Real-life recommendations

I lived for years in Italy, in Rome. Rome is a beautiful place with many food markets—not the central ones found in guidebooks that are full of knock-off Gucci bags (yes, Gucci bags in food markets)—but the ones that are outside the tour bus route, the ones where the locals shop and where farmers sell their products.

Every Saturday we went to see a greengrocer named Marino. We were good customers, real foodies, so he knew that if he recommended good things to us, we'd buy them—even if we had strict plans to buy only what was on our list. The watermelon season was great, the many types of tomatoes offered a fountain of various

flavors, and I'll never forget the taste of the fresh mozzarella. Marino, at times, also recommended that we *not* buy something if it was not top quality, and we trusted him to give us good advice. This is an example of *recommendations*. Marino recommended the same things repeatedly, which is okay with food, but that isn't the case for most other types of products, such as books or movies or music.

When I was younger, before Spotify and other streaming services took over the music market, I liked to buy CDs. I went to a music shop that catered mostly to DJs, and I walked around and gathered a stack of CDs, then found a spot at the counter with a pair of headphones and started listening. With the CDs as context, I had long conversations with the man behind the counter. He checked which CDs I liked (and didn't like) and recommended others based on that. I valued the fact that he remembered my preferences well enough between visits and didn't recommend the same titles to me repeatedly. This is also an example of *recommendations*.

Getting home from work (now that I'm older), I always look in our mailbox to see if we've got mail. Usually, the mailbox is full of advertisements from supermarkets, listing things that are on sale. Typically, the ads show pictures of fresh fruit on one page and dishwasher powder on the next—all things that supermarkets like to recommend that you buy because they claim it's a good offer. These aren't recommendations; they're *advertisements*.

Once a week, the local newspaper is among the mail. The newspaper features a top 10 list of the most watched movies at the theater that week. This is a *non-personalized recommendation*. On television, much thought goes into placing commercials with the right television content. These are *targeted commercials* because it's thought a certain type of people are watching.

In February 2015, Copenhagen Airport officials announced the placement of 600 monitors around the airport to show commercials based on the viewer's estimated age and gender, along with information regarding the destinations at the nearby gates. The age and gender were inferred using cameras and an algorithm. The press release about the advertising provided this description: *"A woman traveling to Brussels wants to see nice watches or an ad for a finance magazine, for example. A family going on vacation might be more interested in ads for sunblock or car rentals."*[1] These are *relevant commercials* or *highly targeted commercials*.

People usually perceive commercials on television or at the airport as a nuisance, but if we go online, the limits to what we consider invasive become a bit different. There could be many reasons for this, which is a whole topic in itself.

The internet is still the Wild West, and although I think that the advertising at the Copenhagen Airport is quite invasive, I also find it irritating when I see advertisements on the internet that are directed at a target group that I'm not part of. To target their commercials, websites need to know a bit about who you are.

[1] For more information, see http://mng.bz/ka6j.

In this and later chapters, you'll learn about recommendations, how to collect information about the recipients of the recommendations, how to store the data, and how to use it. You can calculate recommendations in various ways, and you'll see the most used techniques.

A recommender system isn't only a fancy algorithm. It's also about understanding the data and your users. Data scientists have a long running discussion on whether it's more important to have a super-good algorithm or to have more data. Both have flipsides; super algorithms require super hardware and lots of it. More data creates other challenges, like how to access it fast enough. Going through this book you'll learn about the tradeoffs and get tools to make better decisions.

The previous examples are meant to illustrate that commercials and recommendations can look similar to the user. Behind the screen, the intent of the content is different; a *recommendation* is calculated based on what the active user likes, what others have liked in the past, and what's often requested by the receiver. A *commercial* is given for the benefit of the sender and is usually pushed on the receiver. The difference between the two can become blurry. In this book, I'll call everything calculated from data a recommendation.

1.1.1 Recommender systems are at home on the internet

Recommenders are most at home on the internet because this is where you can not only address individual users but can also collect behavioral data. Let's look at a few examples.

A website showing top 10 lists of the most sold bread-making machines provides *non-personalized* recommendations. If a website for home sales or concert tickets shows you recommendations based on your demographics or your current location, the recommendations are *semi-personalized*. Personalized recommendations can be found on Amazon, where identified customers see "Recommendations for you." The idea of the personalized recommendation also arises from the idea that people aren't only interested in the popular items, but also in items that aren't sold the most or items that are in the long tail.

1.1.2 The long tail

The long tail was coined by Chris Anderson in an article in *Wired* magazine in 2004, which was expanded into a book published in 2006 (Hyperion).[2] In the article, Anderson identified a new business model that's frequently seen on the internet.

Anderson's insight was that if you've a brick-and-mortar shop, you've a limited amount of storage and, more importantly, a finite space to show products to your customers. You also have a limited customer base because people have to come to your shop. Without these limitations, you don't have to sell only popular products as with the

[2] For more information on the magazine article, see https://www.wired.com/2004/10/tail/. For information on the book, see https://en.wikipedia.org/wiki/The_Long_Tail_(book).

usual commerce business model. In brick-and-mortar shops, it's considered a losing strategy to stock non-popular products because you need to store many items that might never sell. But if you've a web store, you can store an infinite number of products because rental space is cheap or, if you sell digital content, it doesn't take up any space at all, costing little or nothing. The idea behind the long-tail economy is that you can profit by selling many products, but only a few of each, to many different people.

I'm all for diversity, so I think it's great to have a huge catalog of products, but the question that's difficult to answer is how do users find what they want? This is where recommender systems make their entrance. Because these systems help people find those diverse things that they wouldn't otherwise know existed.

On the web, because Amazon and Netflix are considered the giants both in content and in recommendations, these companies are used in numerous examples throughout this book. In the following section, you'll take a closer look at Netflix as an example of a recommender system.

1.1.3 The Netflix recommender system

As you likely know, Netflix is a streaming site. Its domain is that of films and TV series, and it has a continuous flow of available content. The purpose of Netflix's recommendations is to keep you interested in its content for as long as possible and to keep you paying the subscription fee month after month.

The service runs on many platforms, so the context of its recommendations can differ. Figure 1.1 is a screenshot of Netflix from my laptop. I can also access Netflix from my TV, my tablet, and even my phone. What I want to watch on each platform varies—I never watch an epic fantasy film on my phone, but I love them on TV.

Let's begin this walk-through by looking at that startup page. The front page is constructed as a panel containing rows with subjects such as Top Picks, Drama, and Popular on Netflix. The top row is dedicated to what's on my list. Netflix loves this list because it indicates not only what I've watched and what I'm watching now, but also what I (at least at one point) have shown an interest in watching.

Netflix wants you to notice the following row because it contains the *Netflix Originals*—the series that are produced by Netflix. These are important to Netflix for two reasons, both financial:

- Netflix has spent big money to produce original content and the programs are, in most cases, found only on Netflix.
- Netflix must pay content owners when users watch their content. If that owner is Netflix, not only does it save them money, it puts money in their pocket.

The last point also illustrates something to consider: even if everything is personalized on the page, the fact that the Netflix Originals are on the second row probably isn't a result of me watching them, but rather a pursuit of an internal business goal.

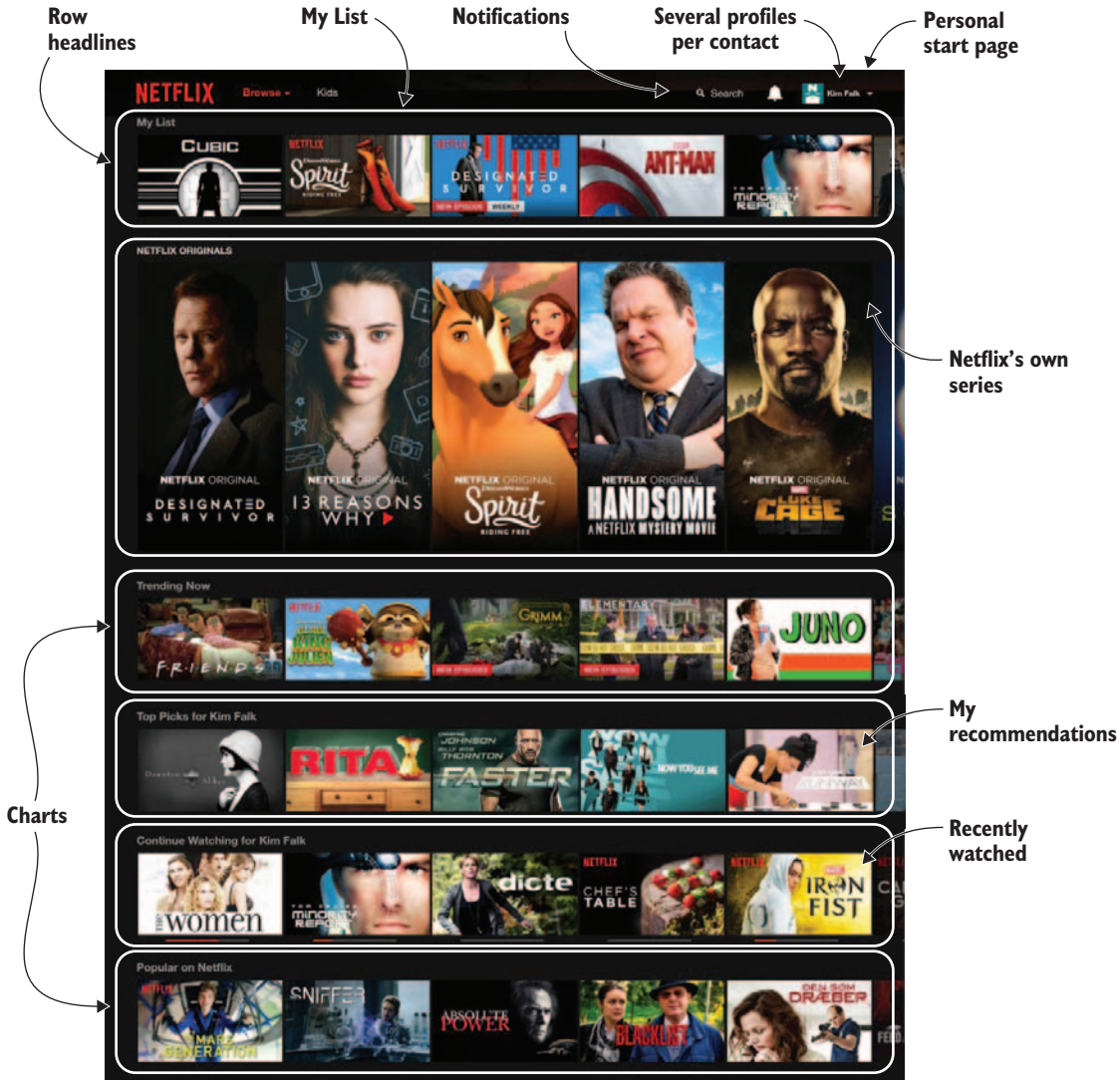

Figure 1.1 The Netflix start page (before it changed the layout)

Charts and trends

Next is the Trending Now list. Trending is a loose term that can mean many things, but here it includes content that's popular within a short period. The bottom row, Popular on Netflix, also has to do with popularity but over a longer period, maybe a week. Trends and charts will be discussed in detail in chapter 5.

RECOMMENDATIONS

The fourth row is the list of Top Picks for me, which match my profile. This list contains what most people would call recommendations. It shows what the Netflix recommender system predicts what I'd like to watch next. It looks almost right. I'm not into bloody, gory movies, and I'd rather not see any dissections of bodies at all. Not all the suggestions are to my liking, but I assume that it's not only my taste that Netflix uses to build this list. The rest of my household also watches content using my profile at times. *Profiles* are Netflix's way of letting the current user indicate who's watching.

Before introducing profiles, Netflix aimed its recommendations at a household rather than one person.[3] It tried to always show something for mom, dad, and children. But Netflix has since dropped that, so now my list doesn't include any children's shows. But even if Netflix is using personal profiles, I think it imperative to consider who's watching—not only the person with the profile, but also anyone else. I've heard rumors that other companies are working on solutions enabling you to tell the system that other people are watching too. This is to allow the service to deliver recommendations fitting all members of the audience. To date, I haven't seen any in play.

Microsoft Kinect could recognize people in front of the TV by using face/body recognition. Microsoft took it a step further by identifying not only household members, but also other people from its full catalog of users, allowing Kinect to recognize users when they're visiting other homes. Although a sign of audience recognition, Kinect for Xbox One was discontinued in October 2017, representing the end of the Kinect product line.

ROWS AND SECTIONS

Back to the Top Picks of Netflix. You can find more details on the content by hovering your mouse over one of the suggestions. A tooltip appears with a description (see figure 1.2) and a predicted rating, which is what the recommender system estimates I'd rate this content. You might expect that the recommendations in the Top Picks all have a high rating, like the one in figure 1.1, but looking through the recommendations, you can find examples of items with a low predicted rating, as shown in figure 1.3.

The ways of the Netflix recommender are many, so there are numerous possible explanations as to why Netflix recommends an item that it predicts I won't rate highly. One reason could be that Netflix is aiming for diversity over accuracy. Another reason could be that even if I won't rate a movie maximum stars, it might still be something that I'm in the mood to watch. This is also the first hint that Netflix doesn't put much value on ratings.

The titles of each row are different; some are of the type Because You Watched Suits. These lines recommend things that are similar to *Suits*. Other rows are genres such as *Comedies*, which, curiously enough, contains comedies. You could say that the row titles are also a list of recommendations; you could call these *category recommendations*.

[3] "Netflix Recommendations: Beyond the 5 stars (Part 1)," http://mng.bz/bG2x.

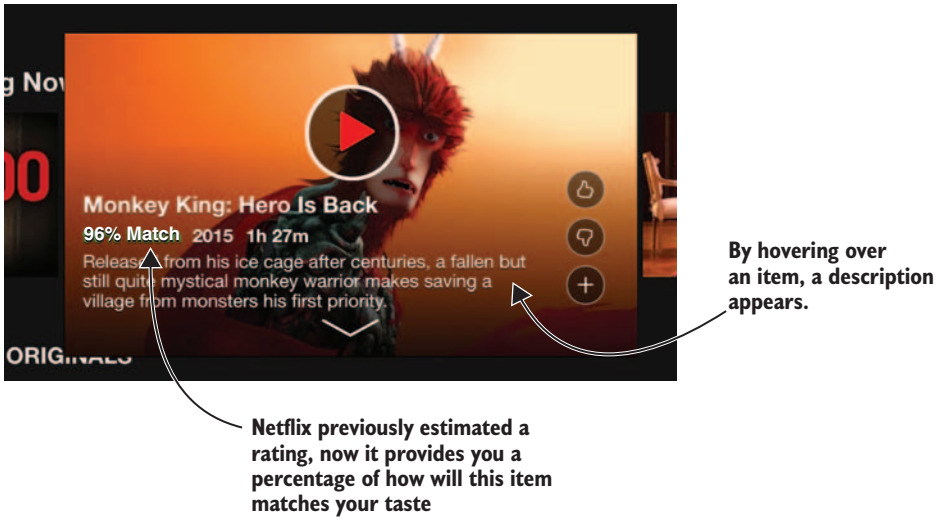

Figure 1.2 A Netflix Top Pick with a predicted match

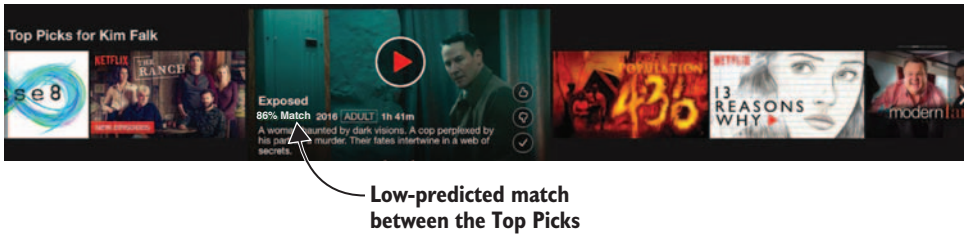

Figure 1.3 A Netflix Top Pick with a low predicted rating

This could be the end of the story, but then you'd miss the most important part of the Netflix personalization.

RANKING

Each of the row headlines describes a set of content. This content is then ordered according to a recommendation system and presented in order of relevancy or *rank*, starting from the left as illustrated in figure 1.4.

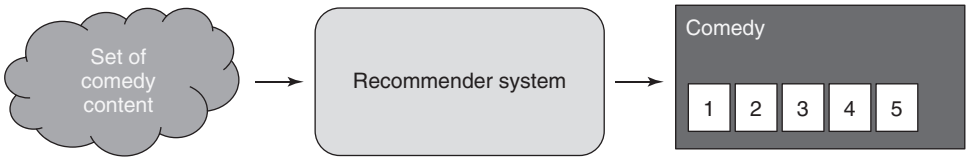

Figure 1.4 Each Netflix row is ordered by relevance.

Even in My List, which contains the content I've selected myself, the content is ordered according to the recommender system's estimate of its relevance for me. I added the screenshot in figure 1.1 yesterday. Today my list has a new order, as shown in figure 1.5.

Figure 1.5 Netflix orders my list by relevancy.

The Netflix recommender system also tries to recommend content that's relevant at a specific time or in a particular context. For example, Sunday mornings might be more for cartoons and comedies, whereas evenings might be more for "serious" watching of a TV series such as *Suits*.

Another row that might be surprising is Popular on Netflix, which shows content that's popular right now. But Netflix doesn't say that the most popular item is the one all the way to the left. Netflix finds the set of most popular items and then orders them according to what you consider most relevant now.

BOOSTING

A point to ponder is why Netflix has ranked the show *Designated Survivor* high in My List, considering that I'm already watching it. But Netflix had a notification indicating that a new season of *Designated Survivor* is out. This could explain why this show appears.

Boosting is a way for companies to put a finger on the scale when suggestions are calculated, and Netflix wants me to notice *Suits* because it's new content, meaning it has a freshness value. Netflix boosts content based on freshness; *freshness* can mean that it's new or it's been mentioned in the news. Boosting is covered in more detail in chapter 6 because it's something that many site holders request as soon as the system is up and running.

> **NOTE** There's a machine-learning algorithm family called boosting, but what I'm referring to here is something different.[4]

[4] For more information, see https://en.wikipedia.org/wiki/Boosting_%28machine_learning%29.

SOCIAL MEDIA CONNECTION

For a short period of time, Netflix also tried to use social media data.[5] Back then, you'd find something like what's shown in figure 1.6 on your Netflix page.

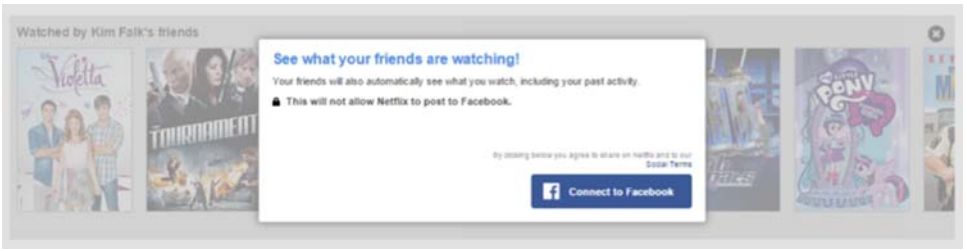

Figure 1.6 Netflix wants to know what my friends are watching.

Netflix encouraged you to enable Facebook Connect, thereby allowing Netflix access to your list of friends as well as other information. One of the advantages for Netflix was that it was able to find your friends and make social recommendations based on what they liked. Connecting with Facebook could also make watching films a much more social experience, which is something that many media companies are exploring.

In this day and age, people don't sit down to watch films passively. They multitask, watching a movie while sitting with a second device (such as a tablet or a smartphone). What you're doing on the second device can have a large influence on what you watch next. Imagine that after you watch something on Netflix, a notification pops up on your phone that one of your friends liked a film, and presto, Netflix recommends that as the next thing to watch.

This social feature was, however, removed in the 2015-2016 timeframe, with the argument that people weren't happy with sharing their films with Facebook's network. In the words of Neil Hunt, the Chief Product Officer at Netflix, "It's unfortunate because I think there's a lot of value in supplementing the algorithmic suggestions with personal suggestions."[6]

TASTE PROFILE

With a page that's built almost entirely based on suggestions, it's a good idea to provide as much input as possible on your tastes. If Netflix doesn't have a clear sense of your taste, it can be hard for you to find what you want to watch.

In 2016 Netflix had options that helped users build their profiles. The Taste Profile menu, shown in figure 1.7, enabled you to rate shows and movies, to select genres by saying how often you felt like watching, for example, *Adrenaline Rush* content as illustrated in figure 1.8, or to check whether your ratings matched your current opinions.

---

[5] "Get to know Netflix and its New Facebook Integration," http://mng.bz/6yHM.
[6] "It's your fault Netflix doesn't have good social features," http://mng.bz/jc7M.

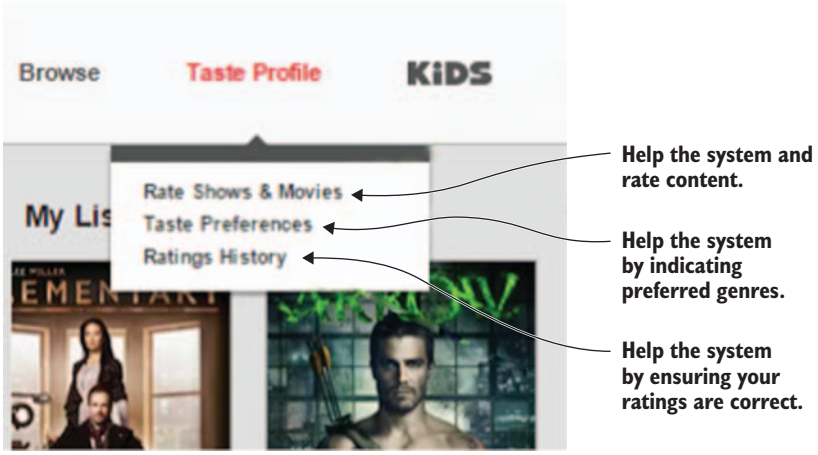

Figure 1.7 Example of how the Netflix taste profile looked in 2015

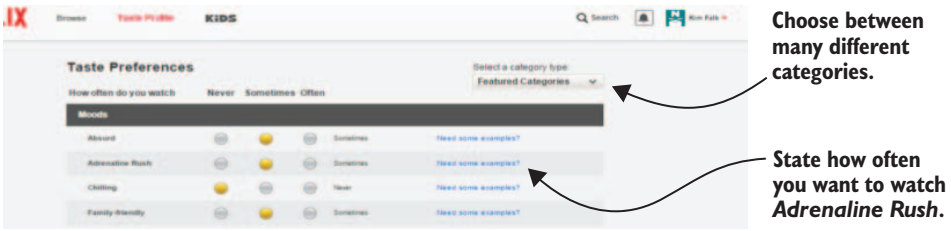

Figure 1.8 The Netflix Taste Preferences menu

The manually inputted Taste Preferences enable Netflix to provide better suggestions. Asking the user for help with the taste profile is a method often used to allow the system to give suggestions to new users. But, as with so many things, there's often a difference between what users say they like and what they indeed like.

Seeing taste preferences is usually the first step in getting to know a user. And as the user uses the system more, Netflix was able to collect usage data, which is often more trustworthy. Netflix has now removed this feature.

1.1.4 Recommender system definition

To be sure we're all on the same page, table 1.1 provides several definitions.

Table 1.1 Recommender system definitions

Term	Netflix example	Definition
Prediction	Netflix guesses what you'll rate an item.	A prediction is an estimate of how much the user would rate/like an item.
Relevancy	Orders all rows on the page (for example, Top Picks and Popular on Facebook) according to applicability.	An ordering of items according to what's most relevant to the user right now. Relevance is a function of context, demographics, and (predicted) ratings.
Recommendation	Top Picks for me.	The top *N* most relevant items.
Personalization	The row headlines in Netflix are an example of personalization.	Integrates relevancy into the presentation.
Taste profile	See figure 1.8.	A list of characterizing terms coupled with values.

With these definitions in place, we can finally define a *recommender system*.

Definition: recommender system

A *recommender system* calculates and provides relevant content to the user based on knowledge of the user, content, and interactions between the user and the item.

With this definition in place, you might think that you've figured it all out. But let's go through an example of how a recommendation could be calculated and how it would work. Figure 1.9 shows how Netflix might produce my Top Picks row. Here are the steps of how Netflix might calculate my Top Picks:

1. A request for the Top Picks list is received.
2. The server calls the recommendation system, which consists of a pipeline of methods. This step is called *retrieve candidate items*. It retrieves the items from the catalog database that are most similar to the current user's taste.
3. The top five items (normally it could be 100 items or more) are piped into the next pipeline step, which is to calculate prediction.
4. Prediction is calculated using the user preferences retrieved from the user database. It's likely that the calculation will remove one or more items from the list due to a small predicted rating. In figure 1.9, items C and E are removed.
5. The significant items are output from the calculated prediction, now with a predicted rating added to them. The result is piped into an order-by-relevance process.

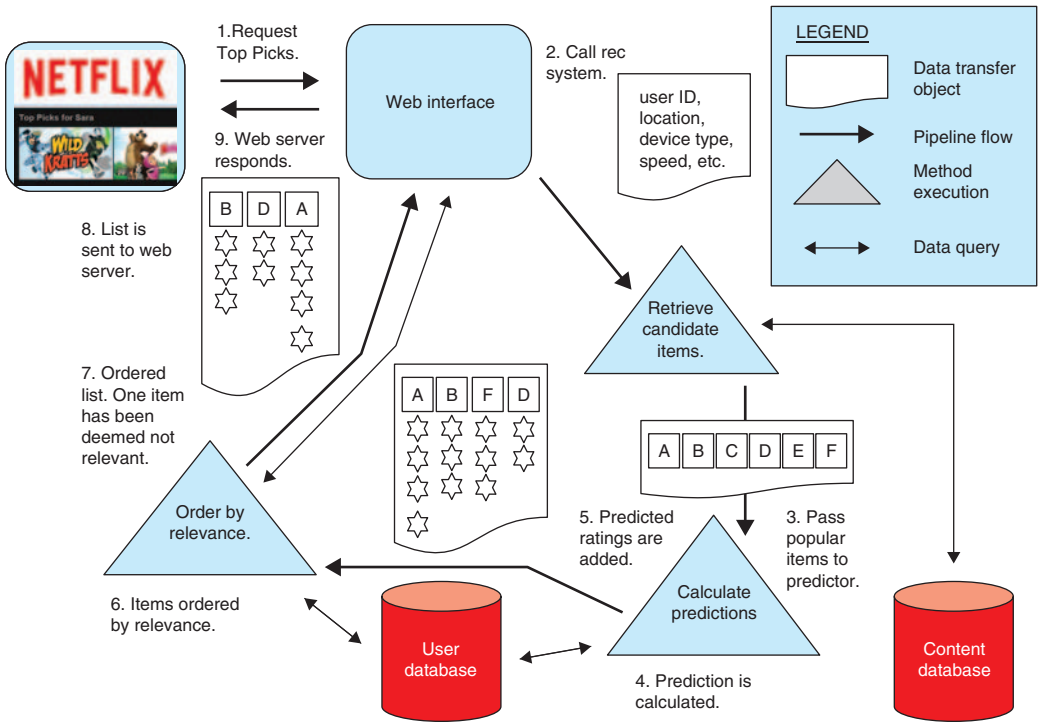

Figure 1.9 How Netflix Top Picks might be calculated

6. The revelant items are ordered according to the user's taste, context, and demographics. The process might even try to add as much diversity to the result as possible.
7. The items are now ordered by relevance. Item *F* was removed because the relevance calculations showed that it wouldn't be relevant for the end user.
8. The pipeline returns the list.
9. The server returns the result.

Looking at figure 1.9, it's evident that there are many aspects to consider when working with recommender systems. The preceding pipeline is also missing the parts of collecting the data and building the models. Most recommender systems try to use the data shown in figure 1.10 in one way or another.

Figure 1.9 also illustrates another fact to take into consideration: the rating prediction is only a part of a recommendation system. Other things can also play an important role in what your system should display to the user. A big part of this book is about predicting ratings, and that's important, even if I made it sound like something negible here.

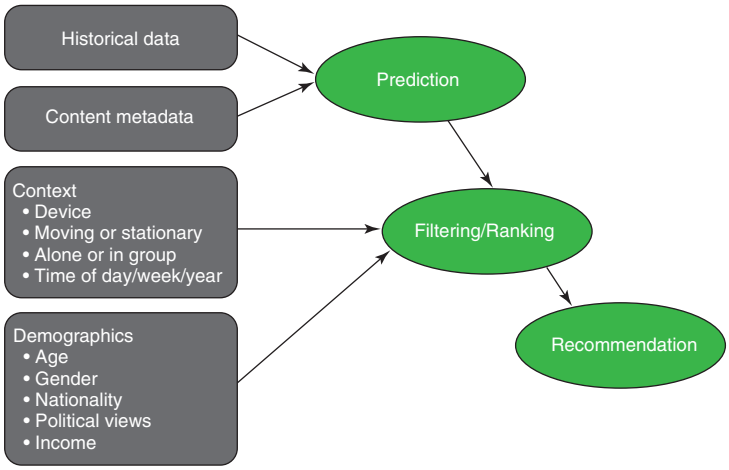

Figure 1.10 Data can potentially be used as input data for a recommender system.

1.2 Taxonomy of recommender systems

Before starting to implement a recommendation system, it's a good idea to dwell a bit on what kind of recommender system you want to roll out of the garage. A good way to start is by looking at similar systems for inspiration. In this section, you'll learn a framework for studying and defining a recommender system.

In the previous sections, the Tour d'Netflix provided an overview of what a recommender system can do. This section explains a taxonomy to use to analyze recommenders. I first learned about it in Professor Joseph A. Konstan and Michael D. Ekstrand's Coursera course "Introduction to Recommender Systems,"[7] and have found good use for it ever since. *Taxonomy* uses the following dimensions to describe a system: domain, purpose, context, personalization level, whose opinions, privacy and trustworthiness, interfaces, and algorithms.[8] Let's look at each of those dimensions.

1.2.1 Domain

The *domain* is the type of content recommended. In the Netflix example, the domain is movies and TV series, but it can be anything: sequences of content such as playlists, best ways to take e-learning courses to achieve a goal, jobs, books, cars, groceries, holidays, destinations, or even people to date.

The domain is significant because it provides hints on what you'd do with the recommendations. The domain is also important because it indicates how bad it is to be wrong. If you're doing a music recommender then it isn't that bad if you recommend

[7] For more information, see www.coursera.org/learn/recommender-systems-introduction/.

[8] The taxonomy concept first appeared in *Word of Mouse: The Marketing Power of Collaborative Filtering* by John Riel and Joseph A. Konstan (Business Plus, 2002).

music that isn't spot on. If you're recommending foster parents to children in need, then the cost of failure is quite high. The domain also dictates if you can recommend the same thing more than once.

1.2.2 Purpose

What is the purpose of the Netflix site, both for the end user and for the provider? For end users, the use of Netflix recommendations is to find relevant content that they want to watch at that specific time. Imagine that you didn't have any ordering or filtering. How would you ever find anything in the Netflix catalog when it has more than 10,000 items? And the purpose for the provider (in this case, Netflix) is ultimately to make customers pay for the subscription month after month by providing content they want to watch, right at their fingertips.

Netflix considers the amount of content viewed as a deciding factor in how they're doing. Measuring something else instead of your direct goal is called using a *proxy goal*. Using a proxy goal is something you should be careful about because it can inadvertently end up measuring other effects than what you wanted—more time spent on the Netflix platform could mean frustrated customers who search and search without finding what they're looking for, or they may have found it, but the site keeps stalling.[9]

Behind the scenes, there might also be considerations to balance things in such a way that Netflix pays the least money possible for what you're watching. Netflix probably pays less to offer 10-year-old episodes of *Friends* than a newer series or, even better, a Netflix original series where they don't have to pay a license fee to anybody.

A purpose could also be to give information or to help or educate the user. In most cases, however, the purpose is probably to sell more.

What type of customers would you rather serve: consumers who arrive once and expect good recommendations or loyal visitors who create profiles and return on a regular basis? Will the site be based on automatic consumption (for example, the Spotify radio station, which keeps playing music based on a song or a single artist)?

1.2.3 Context

The *context* is the environment in which the consumer receives a recommendation. In our example, it can be the device the customer uses to view Netflix or the current location of the receiver, the time of day (or night), and what the consumer is doing. Does the user have time to study the suggestions or is a quick decision needed? The context can also include the weather or even the user's mood!

Consider a search for a cafe on Google Maps. Is the user sitting at an office computer and looking for a good coffee bar, or is the user standing on the street as it starts to rain? In the first scenario, the best response would identify good quality cafes in a wider radius; in the second scenario, recommendations would ideally contain only the

[9] I recommend *Weapons of Math Destruction* by Cathy O'Neil (Broadway Books, 2016) if you want to know more about how wrong things can go when you use proxy goals.

nearest place to drink coffee while the rain passes. Foursquare is an example of an app where you can find cafes. We'll look at Foursquare in chapter 12.

1.2.4 Personalization level

Recommendations can come at many personalization levels, from using basic statistics to looking at individual user data. Figure 1.11 illustrates these levels.

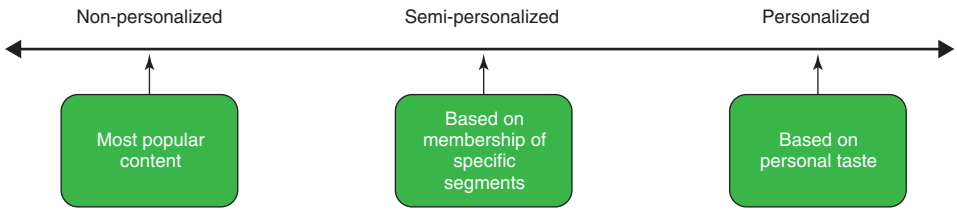

Figure 1.11 Personalization levels

NON-PERSONALIZED

A list of the most popular items is considered a *non-personalized recommendation*: the chances are that the current user might like the same items as most others do. Non-personalized recommendations also include showing things ordered by date, such as showing the newest items first. Everyone who interacts with the recommender system receives the same list of recommendations. And they can also include when a cafe suggests drinks Friday afternoon, cappuccinos in the morning, but brunch on weekend mornings.

SEMI/SEGMENT-PERSONALIZED

The next level of recommendations divides users into groups—the *semi/segment personalized recommendations*. You can segment groups of users in many ways: by age, by nationality, or by distinct patterns such as business people or students, car drivers or bicycle riders.

A system selling concert tickets, for example, recommends shows based on the user's country or city. Here's another case: if a user is listening to music on a smartphone, the system might try to deduce whether the device is moving or not. If it is moving, the person might be exercising or they might be driving or cycling. If the device is stationary, the consumer may be sitting on a sofa at home and the appropriate music might be different.

This recommender system doesn't know anything personal about you, only you as a member of a group or segment. Other people who fit into the same group will get the same recommendations.

PERSONALIZED

A *personalized recommendation* is based on data about the current user that indicates how the user has interacted with the system previously. This generates recommendations specifically for this user.

Most recommender systems also use segments and popularity when creating personalized recommendations. An example of a personalized recommendation is Amazon's Recommended for You. The Netflix starting page is an extreme example of personalized recommendations.

Usually, a site applies various types of recommendations. Only a few sites, such as Netflix, offer everything personalized. On Amazon, you'll also find Most Sold Items, which is nonpersonalized, as well as the Customers Who Bought This Also Bought This list, which provides *seeded recommendations*. These are recommendations based on a seed, which could be the current item that a user is viewing.

1.2.5 Whose opinions

Expert recommenders are manual systems whose experts recommend good wines, books, or similar. These systems are used in areas where it's generally accepted that you need to be an expert to understand what's good.

The days of expert websites are mostly over, however, so the *whose opinions* parameter isn't used much nowadays. Almost all sites use the opinions of the masses. They say that there's no rule without an exception; a few expert sites still remain. An example is the sommelier's recommendation on the wine site called www.vivino.com, shown in figure 1.12. Vivino is turning to recommender systems to recommend wines as well. Vivino added the recommender system to their app in 2017 to help users find new wines to taste based on their rating history.[10]

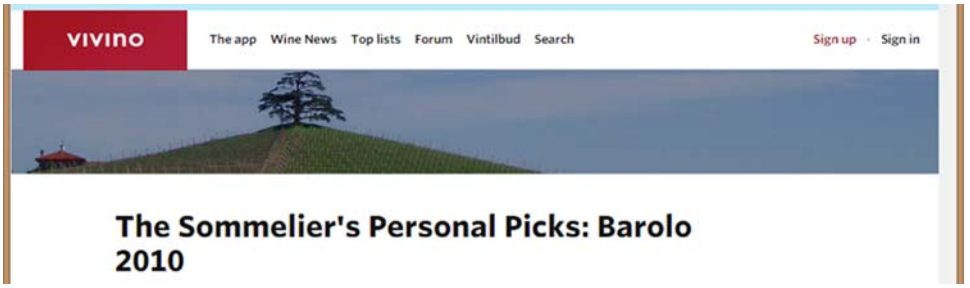

Figure 1.12 Vivino.com provides expert wine recommendations (the recommendations are omitted to save space).

1.2.6 Privacy and trustworthiness

How well does the system protect users' privacy? How is the collected information used? For example, in Europe, it's common to pay money into a pension, which is handled by a bank. Often these banks offer different kinds of retirement savings plans. A system that recommends these should have strict rules for privacy. Imagine

[10] For more information, see http://mng.bz/1jFR.

filling in an application for a retirement savings plan and describing that you've back problems, and a minute later receiving a phone call from a chiropractor with great offers to handle your exact problem. Or even worse, you buy a special bed for people with back problems, and an hour later you receive an email that your health insurance premium has gone up.

Many people consider recommendations as a form of manipulation because they present choices that customers are more likely to pick than if they were offered a random selection. And most shops are trying to sell more, so the fact that stores that use recommendations to sell more makes people think they're being manipulated. But if that means watching a film that would entertain rather than bore, then I say it's okay. Manipulation is more about the *motive* for showing a particular item rather than the *act* of showing it. If you've recommended inappropriate and non-optimal medicine because the vendor buys the website owner better dinners, then that's manipulation, which should be frowned upon.

When the recommendation system starts performing and an increase in business is measured, many might find it tempting to inject vendor preferences, overstocked items, or maybe preferences for which brand of pills customers buy. Beware: if customers start feeling manipulated, they'll stop trusting your recommendations and eventually find what they need somewhere else.

> *The moment that recommendations have the power to influence decisions, they become a target for spammers, scammers, and other people with less-than-noble motives for influencing our decisions.*
>
> —Daniel Tunkelang[11]

Trustworthiness indicates how much the consumer trusts recommendations instead of considering them as commercials or attempts at manipulation. In the Netflix example, I talked about how predictions can be discouraging for users if the estimated prediction is far off the user's actual rating. This is about trustworthiness. If the user takes the suggestions seriously, the system is trustworthy.

1.2.7 Interface

The *interface* of a recommender system depicts the kind of input and output it produces. Let's look at each.

Input

Netflix once enabled users to enter likes and dislikes by rating content and adding preferences on genres and topics. This data can be used as input to a recommender system.

The Netflix example uses *explicit input*, where you, the consumer, manually add information about what you like. Another form of input is *implicit*, where the system

[11] For more on the role taste and trust play in recommendations, see www.linkedin.com/pulse/taste-trust-daniel-tunkelang.

tries to deduce taste by looking at how you interact with the system. Chapter 4 handles feedback in more detail.

OUTPUT

Types of *output* can be predictions, recommendations, or filtering. For example, Netflix outputs recommendations in many ways. It estimates predictions, provides personalized suggestions, and shows popular items, which normally is in the form of a top 10 list (but Netflix even personalizes that).

If the recommendations are a natural part of the page, it's called an *organic presentation*. The rows shown on Netflix are an example of organic recommendations: Netflix doesn't indicate that these are recommendations; they're an integral part of the site.

The examples illustrated in figure 1.13 are nonorganic. Hot Network Questions uses a form of non-personalized recommendations by not explicitly stating that what's shown. Amazon displays nonorganic personalized recommendations in its Recommended for You list, and the *New York Times* employs nonorganic recommendations showing the most emailed articles.

Figure 1.13 Examples of nonorganic, non-personalized recommendations: Hot Network Questions from Cross Validated, Most Emailed from the *New York Times*, and a personalized Recommended for You list from Amazon

Certain systems explain the recommendations. Recommenders with that ability are called *white-box recommenders*; those that don't are called *black-box recommenders*. Figure 1.14 shows examples of each. The distinction is important to consider when choosing an algorithm because not all provide a clear path back to the reasons for a prediction.

Deciding whether you want to produce a white-box or black-box recommender can put constraints on which algorithms you use. The more your system needs to explain, the simpler the algorithm. Often you can consider the decision as shown in figure 1.15. The better the quality of the recommendation, the more complex and the harder to show explanations. This problem is known as *model accuracy-model interpretation trade-off*.

I once worked on a project where extreme emphasis was placed on explainability and quality. To solve this, we had to build another algorithm on top of our recommender system to allow for good quality recommendations while also having a system that connected the evidence with the result.

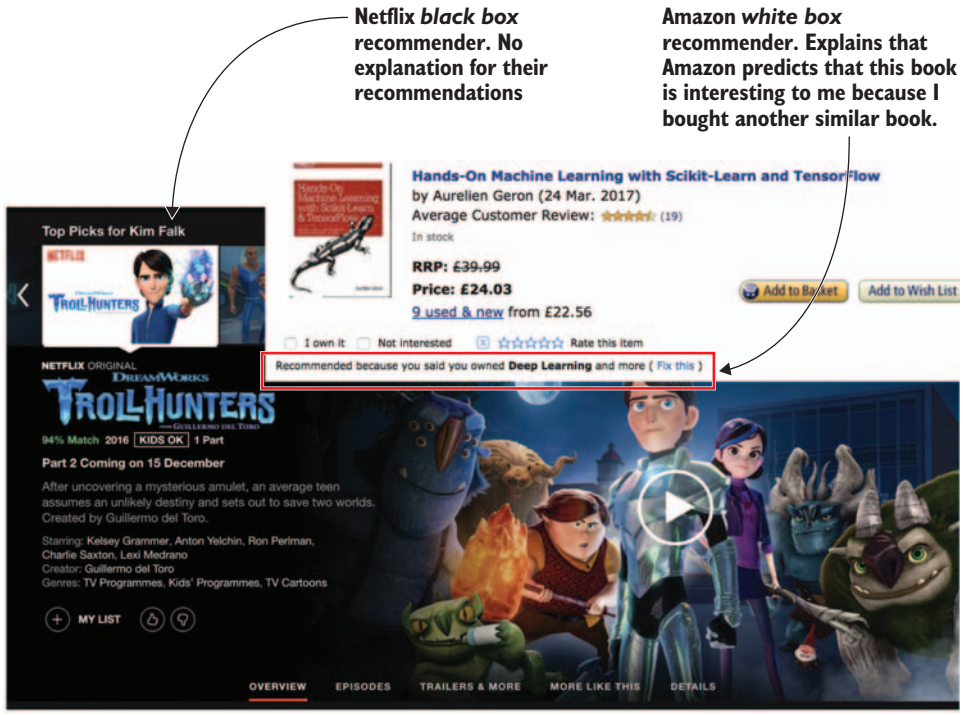

Figure 1.14 Black-box (from Netflix) and white-box (from Amazon) recommendations

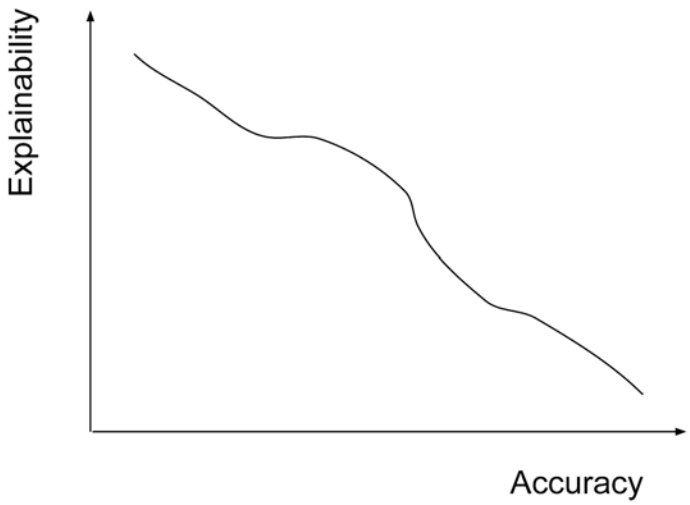

Figure 1.15 Explainability vs. quality of recommendations

Recommender systems have become extremely common in recent years, so there are many examples to look at. Often the recommender systems are implemented for movies, music, books, news, research articles, and most products in general. But recommender systems also have a place in many other areas, such as financial services, life insurance, online data, job searches, and in fact, everywhere there are choices to be made. This book primarily uses websites as examples, but there's no reason not to work in other platforms.

1.2.8 Algorithms

A number of algorithms are presented in this book. The algorithms fall into two groups, and they depend on the type of data you use to make your recommendations. Algorithms that employ usage data are called *collaborative filtering*. Algorithms that use content metadata and user profiles to calculate recommendations are called *content-based filtering*. A mix of the two types is called *hybrid recommenders*.

COLLABORATIVE FILTERING

Figure 1.16 illustrates one way of doing collaborative filtering. The outer set is the full catalog. The middle set is a group of users who have consumed similar items. A recommender system recommends items from the smaller, front-most set, assuming that if users liked the same things as the current user, then the current user will also like other items this group has consumed. The group is identified by the overlap between what the individual users have liked and what the current user liked. Then the gap of content, which the current user is missing, will be recommended (the part of the middle circle that isn't covered by the circle representing the current user's likes).

Many ways exist to calculate collaborative filtering recommendations. You'll see a simple version in chapter 8 and a not-so-simple one in chapter 11, where we talk about matrix factorization algorithms.

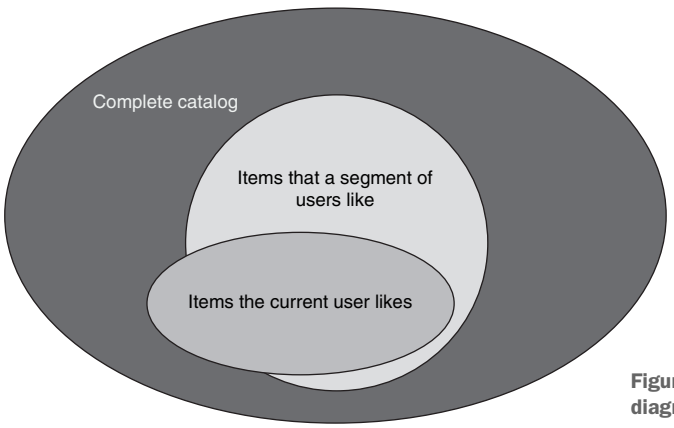

Figure 1.16 Collaborative filtering diagram

CONTENT-BASED FILTERING

Content-based filtering uses the metadata you have on the items in your catalog. Netflix uses descriptions of its movies, for example.

Depending on the specific algorithm, the system can calculate recommendations either by taking the items the user has liked and finding similar content, by comparing the items and user profiles, or, if there's no user involved, by finding similar content between items. When there's a user profile, the system calculates a profile for each user that contains categories of the content. If Netflix used content-based filtering, it could create a user profile of genres like thrillers, comedies, drama, and new films, and give values to them all. Then a film gets recommended if it has similar values as the user.

Here's an example. User Thomas likes *Guardians of the Galaxy*, *Interstellar*, and the TV series *Game of Thrones*. Each is rated according to a five-point system. Table 1.2 shows one way of looking at the three selections.

Table 1.2

Movies and TV	Sci-Fi genre	Adventure genre
Interstellar	3	3
Game of Thrones	1	5
Guardians of the Galaxy	5	4

Based on this information, you build a profile of Thomas indicating Sci-Fi: 3, Adventure: 4. To find other films to recommend, you look through the catalog to find films similar to Thomas' profile.

HYBRID RECOMMENDER

Both collaborative filtering and content-based filtering have strengths and weaknesses. Collaborative filtering needs much feedback from the users to work properly, while content-based filtering needs good descriptions of the items. Often recommendations are produced as a mix of the output from the two types of algorithms we talked about previously, plus other types of input, which could be the distance from a place or the time of day.

1.3 Machine learning and the Netflix Prize

A recommender system is about predicting what content a user needs right now. You can predict this in many ways. Building recommender systems has become a multidisciplinary sport that takes advantage of computer science fields such as machine learning, data mining, information retrieval, and even human-computer interaction. Machine learning and data-mining methods enable the computer to make predictions by studying examples of what it should predict; consequently, recommendations can be constructed by using these prediction functions.

Many recommender systems are centered around machine-learning algorithms to predict user ratings of items or to learn how to correctly rank items for a user. One reason that the field of machine learning is growing is that people want to solve the recommender system problem. The aim is to implement algorithms that enable computers to suggest our secret desires, even before we know them ourselves.

Many claim that the catalyst for this interest in applying machine learning to recommender systems was the famous Netflix Prize. The *Netflix Prize* was a competition hosted by Netflix that offered $1,000,000 to anyone who could come up with an algorithm that improved their recommendations by 10%. The competition began in 2006, and it took almost three years for somebody to win it. In the end, it was a hybrid algorithm that won. You'll recall that a hybrid algorithm runs several algorithms and then returns a combined result from all of them. You'll learn about hybrids in chapter 11.

Netflix never used the winning algorithm, probably because it was so complicated that the performance hit on the system couldn't justify the improvements. Sadly, we don't have Netflix to play with while learning about recommender systems. Instead, I've implemented a small demo site called MovieGEEKs to show off the things described in this book. The site requires much tweaking before it can be production-ready. Understanding recommender systems is its key purpose.

1.4 The MovieGEEKs website

This book is about how to implement recommender systems. It will provide you with the tools to do that, no matter which platform you want to use for your recommendation system. But to do anything interesting with a recommender system, you need data and to get a feel for how it's working it's not enough to look at numbers.

This book focuses on websites, but that doesn't mean that everything written here doesn't apply to any other type of system. This is a short introduction to the framework in which we'll do our dance.

The MovieGEEKs website (http://mng.bz/04k5) is built using a Django website. I encourage you to download MovieGEEKs and use it as you read through the book because it will help you understand what's going on. The fact that it's a Django site or something else isn't so important; I'll point you to where to look as we work through the examples.

> **Django website and framework**
> If the words *Django web framework* sound strange to you, take a look at the Django documentation at www.djangoproject.com/start/overview/.

You'll download the website once. It contains all the functionality described in this book. Here's the fictional scenario that we'll be following.

Imagine that you have a customer who wants to take his DVD selling online. I imagine an old DVD rental shop that was in Bath, in the United Kingdom, with an owner who wants to try movie selling on the internet. The store sadly no longer exists (see figure 1.17).

The shop was anything but electronic; it was managed with small paper cards and, although you might think that sounds impossible, it all seemed to work! In real life, I don't think the owner would ever have taken his business online, but one of the unique things about this place was that you'd always get superb recommendations. The owner would do a monthly review—expert opinion recommendations—and the people who worked there always knew everything there was to know about films.

I like to think of recommender systems as an attempt to give personal service to people on the Net. The following is a brief description of what the fictive owner wants.

Figure 1.17 The facade of On the Video Front, our fictional business

1.4.1 Design and specification

To get started, you'll need to put down several overall points for the design. The main page of the site should show visitors the following:

- A tiled area of movies
- An overview of each film, without leaving the page
- Recommendations as personal as possible
- A menu containing a list of genres

Each movie should have its own page with details as follows:

- Movie poster
- Description
- Rating

Each category should have a page containing the following:

- Same structure as the homepage
- Recommendations specific to the category

1.4.2 Architecture

You'll use Python and the Django web framework to implement this site. Django lets you split a project into different applications. Figure 1.18 shows a high-level architecture and provides an illustration of which applications will build the site.

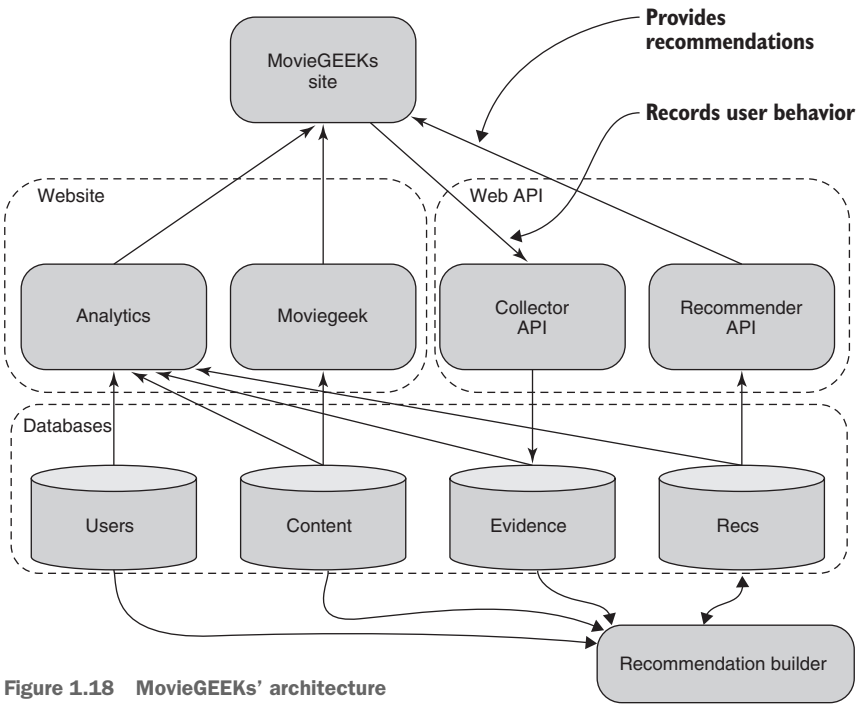

Figure 1.18 MovieGEEKs' architecture

Let's do a quick walk-through:

- *MovieGEEKs*—This is the main part of the site. Here the client logic (HTML, CSS, JavaScript) is placed along with the Python code responsible for retrieving the movie data.
- *Analytics*—The ship's bridge, where everything can be monitored. This part will use data from all the databases. The analytics part is described in chapter 4.
- *Collector*—This handles the tracking of the user behavior and stores it in the evidence database. The evidence logger is described in chapter 2.
- *Recs*—This is the heart of this story and is what will add the edge to this site. It'll deliver the recommendations to the MovieGEEKs site. The recommendations part is described in chapter 5 and the rest of the book.
- *Recommendation builder*—This pre-calculates recommendations, which will provide elaborate recommendations to the user. You'll meet the recommendation builder for the first time in chapter 7.

Each of these components or applications contain exciting data models and features. This will thoroughly entice future visitors.

MovieGEEKs is a movie site mainly because a data set is available that contains a long list of content—movies, users, and ratings. Even more important, the content includes URLs that translate into movie posters, which makes working with it much more fun.

Figure 1.19 shows the MovieGEEKs homepage, or landing page. When the user clicks a movie, a pop-up appears that gives more information and a link to even more details.

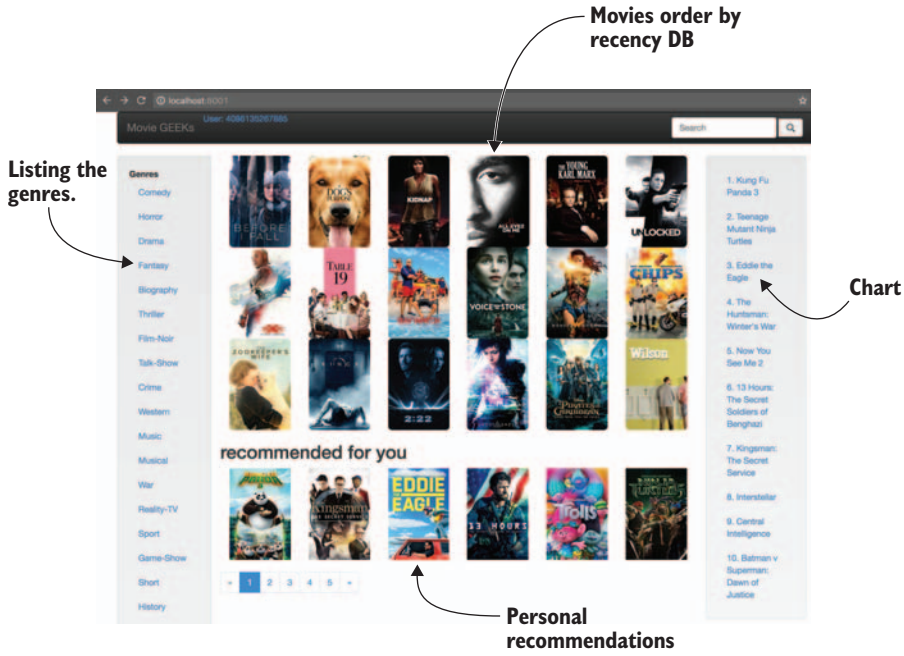

Figure 1.19 The landing page of the MovieGEEKs site

That's it! Simple, but it'll do the trick. Go ahead and download it now. For installation instructions, refer to the readme on GitHub at http://mng.bz/04k5. The MovieGEEKs site uses a data set called MovieTweetings. This data set consists of ratings on movies that were contained in well-structured tweets on Twitter.[12]

1.5 Building a recommender system

Before moving on, let's look at how you'd build a recommender system. Assuming you already have a platform in the shape of a website or an app where you want to add a recommender system, it'd go something like the cycle shown in figure 1.20.

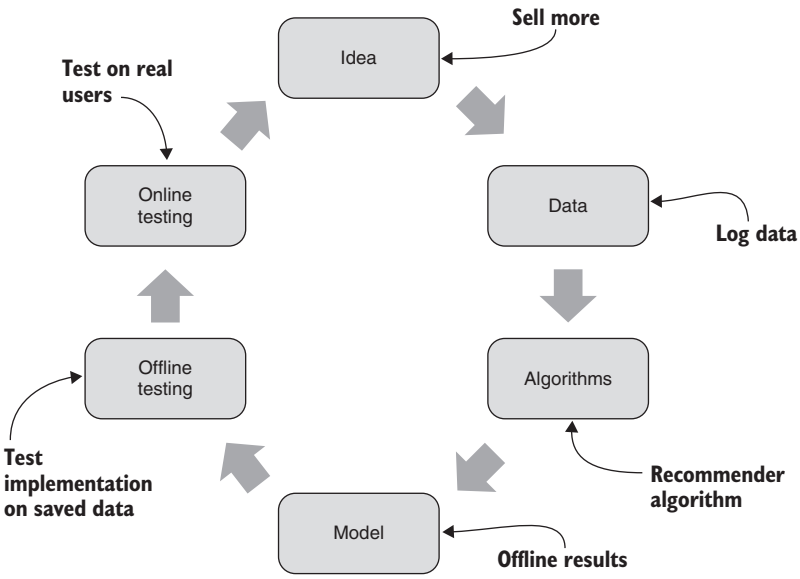

Figure 1.20 A data-driven approach to building a recommender system

Start with an idea that you want to sell more by adding recommenders. You'll collect behavioral data and use that data to build an algorithm, which creates a model when it runs. The model can also be considered as a *function*, which will, given a user ID, calculate recommendations.

You'll try out this model on historical data to see if you can use it to predict a user's behavior. For example, if you have data showing what users bought last month, then you can create the model using the three first weeks of data to see how well the model recommends things that the users bought in the last week of your month of data. It might be better at predicting what the user has bought compared to a baseline recommender system, which can be as simple as a method that returns the most popular items. If it's doing well, you can expose it to part of your users and see if you can track

[12] For more information, see https://github.com/sidooms/MovieTweetings.

improvement. If you see improvement, then it can go into production; otherwise, it's back to the drawing board.

You should now have an idea of what a recommender system is, an understanding of what's needed as input, and what it can produce. Knowing the basis of a recommender system gives you the foundation for chapter 2, which shows how to collect data from users.

Summary

- Netflix uses recommendations to personalize its site and to help users select things they like.
- A recommender system is a common term for many different components and methods.
- A prediction is different than a recommendation. A prediction is about projecting what rating a user would give content, while a recommendation is a list of items that's relevant to the user.
- A recommendation context is what happens around the user (the user's environment) when a recommendation arrives. Items that might not be predicted to have the highest ratings can be recommended if they suit the context.
- The taxonomy described in this chapter is handy when you're looking at other recommender systems or trying to design your own. It's good to go through this taxonomy before starting to implement your own recommender system.

2 User behavior and how to collect it

This chapter invites you to delve into the interesting subject of data collection:

- You'll start by returning to the Netflix site to identify events, which can provide evidence to build a case for what a user likes.
- You'll learn how to build a collector to gather these events.
- You'll learn how a collector can be integrated into a site such as MovieGEEKs to fetch events similar to the ones identified on the Netflix site.
- With a general overview in place and an implementation, you'll step back and analyze general consumer behavior.

Evidence is the data that reveals a user's tastes. When we talk about collecting evidence, we're collecting events and behavior that provide an indication of the user's tastes.

Most books on recommender systems describe algorithms and ways of optimizing them. They start at a point where you already have a large data set to feed your algorithms. You'll use one such data set in the MovieGEEKs site. This data set contains a catalog of movies and ratings from real users. A data set doesn't magically appear. Gathering the right evidence takes work and consideration. It'll also make or break your system. "Garbage in, garbage out," that famous programming saying is also true for recommenders.

Sadly, data that's good for one system might be unsuitable for another. For this reason, we'll have a serious discussion about data that *could* be usable, but I cannot

guarantee that everything described will work in exactly the same way in your environment. In this chapter and throughout the book, we'll look at many examples of how to approach data collection for your own site.

Generally, two types of feedback are produced by users of a system: *explicit* (ratings or likes) and *implicit* (activity recorded by monitoring the user). A user can provide explicit feedback in the form of a certain number of stars, hats, smileys, or any other icon illustrating how much the user likes a product. Usually the scale is between one and five (or one and ten). User ratings are often the first thing people think about when talking about evidence. Later in this chapter, you'll look at ratings, but they're not the only thing that indicates what a user likes.

Ron Zacharski's *A Programmers Guide to Data Mining* presents a great example that illustrates the difference between implicit and explicit evidence.[1] He shows the explicit evidence of a guy named Jim. Jim states he's a vegan and enjoys French films, but in Jim's pocket is a rental receipt for Marvel's *The Avengers* and one for a 12-pack of Pabst Blue Ribbon beer. Which should you use to recommend things? I think it's an easy choice. What do you think Jim wants recommended when he opens the online ordering site for his local takeout place: vegan food or fast food? By collecting user behavior data, you can understand what users like Jim want.

You'll find there's no substitute for good evidence. Let's look at what Netflix could record, and how they could interpret it.

2.1 How (I think) Netflix gathers evidence while you browse

Let's return to Netflix for an example of evidence. Everything on the Netflix landing page is personalized (the row headlines as well as their content), with the exception of the top row, which is an advertisement that everybody probably sees or perhaps only users similar to me. Figure 2.1 shows my personalized Netflix page.

The rows are different for each user; the headlines range from familiar categories such as Comedies and Dramas to highly tailored slices such as Imaginative Time Travel Movies from the 1980s.[2] On my page, the first couple of rows contain recently added content, suggestions, and popular content. On that day, the first personalized row title is Dramas, which indicates that between the genres (or row headlines), Netflix predicts that drama best matches my interests. The Dramas row contains a list of content that Netflix considers interesting to me within that category. Figure 2.2 illustrates the Netflix content shown in my Drama list.

Hovering the mouse over the row, I can make it scroll sideways, showing other content in the Dramas genre. When I scroll over the content of the Dramas row, what

[1] A free download of this book is available at http://guidetodatamining.com/.

[2] This example comes from Netflix. I'd love to know what's inside that category because I'm sure I'd want to watch all the films, but besides the *Back to the Future* films, what could possibly be in there? Look at this article for more details on Netflix's colorful categories at http://mng.bz/xCvj.

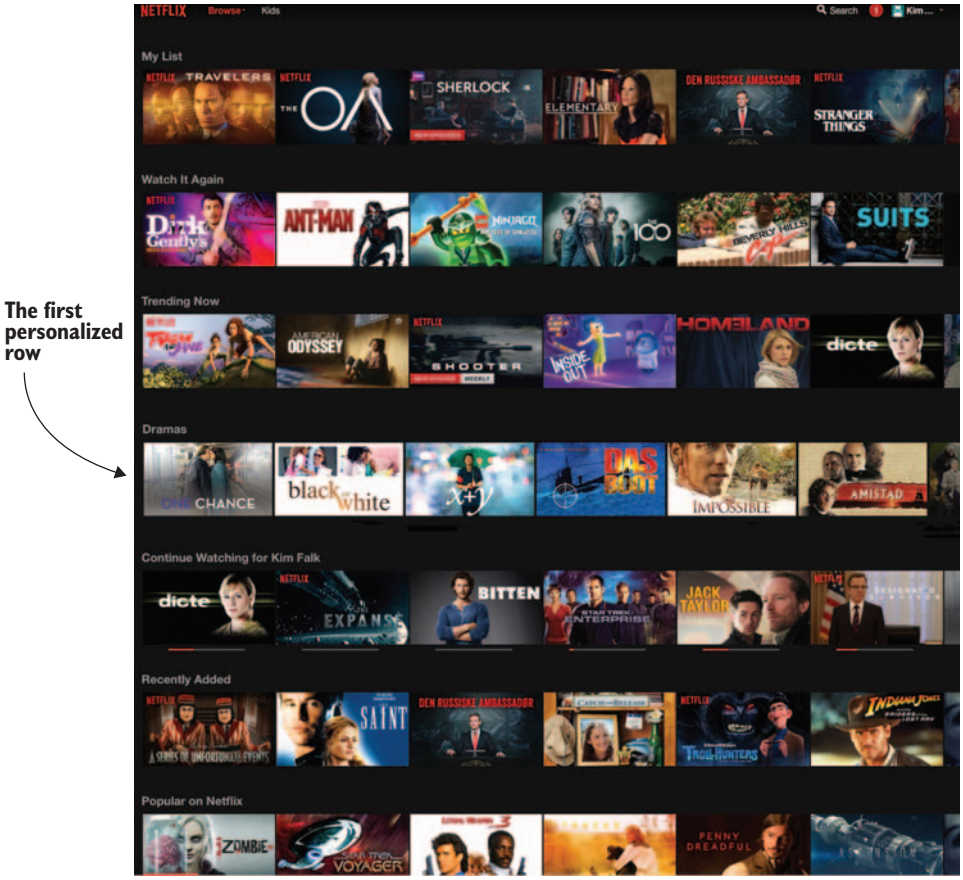

Figure 2.1 My personalized landing page on Netflix. On this day, Drama occupies the top spot.

does this say about me? It could mean that I'm writing a book and trying to make screenshots, but most likely it suggests that I enjoy dramas and want to investigate this list further.

If I see a movie that looks interesting, I can place my mouse over it to view details. Moreover, if those details are intriguing, I can click to go to the films's page. If it still sounds good, I'll either add it to My List or start watching it.

You could say that for any e-commerce site, looking at details means that I'm interested; only the act of watching (and finishing) a film is equivalent to buying. But for a streaming site, this isn't where it ends. If a visitor starts watching a film, it's a positive event, but if they stop after three minutes and never restart the film, then it shows that the visitor didn't like what they saw. If they restart the movie later or within a certain timeframe, it might mean that they liked it even more than when they'd watched it the first time.

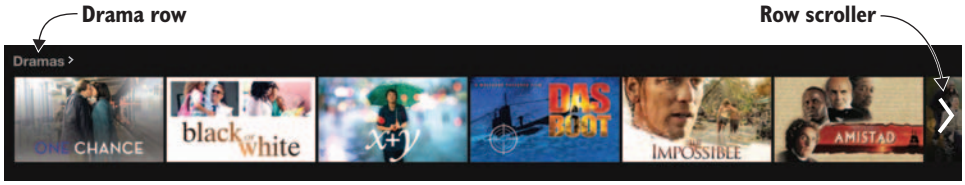

Figure 2.2 Netflix Drama row, where you can either scroll the dramas using the arrow or click the row title and get a complete listing.

OFTEN THE PURPOSE ISN'T WHAT IT SEEMS

Often the purpose of e-commerce sites is to make people buy products, even if the product the customer is buying might not be exactly what they want. That depends on the site's affiliation to the product. If Amazon sells you a poor quality T-shirt, for example, you'd say that the brand of the T-shirt is bad, but you might go back to Amazon and buy another T-shirt. If you buy a T-shirt from, say, a Gap website, and you didn't like the quality, then you'll probably go to another site.

Subscription-based sites are a bit different. Mofibo (https://mofibo.com/), an e-book streaming service available in Danish, Swedish, and Dutch markets, provides recommendations as inspiration and discovery, but with a catch—it's important to Mofibo that the reader knows what kind of book it is before starting to read it. Because Mofibo pays a fee every time a reader opens a new book (not per page, but per book) and, although Mofibo wants you to read as much as possible, it also wants to minimize the number of books it must pay for.

2.1.1 The evidence Netflix collects

Let's try to imagine what goes on behind the curtains at Netflix and what data they collect. Say it's Saturday night at Jimmie's place. Jimmie has Netflix user ID 1234, and after finishing nuking the popcorn in the microwave, he opens Netflix and does the following:

- Scrolls the Drama (ID 2) row
- Hovers the mouse over a movie (ID 41335) to get details
- Clicks to get more details about the movie (ID 41335)
- Starts watching the movie (ID 41335)

While he watches the movie, imagine what happened on the Netflix server. Table 2.1 shows several of the events that could be collected from this user, along with interpretations of what they could mean. Moreover, I've added a column containing an event name to connect table 2.1 to the log described later.

Table 2.1 Examples of evidence from Netflix

Event	Meaning	Event name
Scrolling a themed row	User is interested in the theme, here Dramas.	`genreView`
Placing the mouse over a film to request an overview of content	User is interested in the movie (a drama), thereby showing interest in this category.	`details`
Clicking the film to request the details of content	User is more interested in the movie.	`moreDetails`
Adding the film to My List	User intends to watch the movie later.	`addToList`
Starting to watch the film	User "purchases" the movie.	`playStart`

As table 2.1 illustrates, all these events are evidence to the system because they uncover interests of the user. Table 2.2 shows how the evidence might be recorded by Netflix.

Table 2.2 How Netflix probably logs evidence

userId	contentId	Event	Date
1234	2	`genreView`	2017-06-07 20:01:00
1234	41335	`details`	2017-06-07 20:02:21
1234	41335	`moreDetails`	2017-06-07 20:02:30
1234	41335	`addToList`	2017-06-07 20:02:55
1234	41335	`playStart`	2017-06-07 20:03:01

There'll probably be a long list of other columns such as device type, location, speed, and weather, that can all be used to better understand the user's context. I'd also venture that the number of log events for this scenario would be many more, but let's keep the example simple. The list of events types is probably much longer as well.

Now that you have a general idea what evidence is, you can start looking into the implementation of an evidence collector. An *evidence collector* is used to collect data like that found in table 2.2. To ensure that you're not thinking that this can work only with media streaming sites, let's look at another scenario.

GARDEN TOOLS SITE EXAMPLE

I once had a colleague who spent all his breaks surfing the internet for garden tractors or for anything that ran on gasoline that he could use in a garden. Let's imagine this former colleague had a short break and opened his favorite (imaginary) site called Super Power Garden Tools. He did the following:

- Selected the Garden Tractor category
- Clicked a green monster that can pull up trees

- Clicked Specifications to see the size of the trees it can pull up
- Bought the green monster

These events were the same as picking a movie; the importance of buying an expensive product might be more significant than watching a movie, but I hope you get the picture.

2.2 Finding useful user behavior

Sites with high user involvement enable the site owner to collect large amounts of relevant data, whereas sites with mostly one-time visitors need to focus on relationships among the content instead. Don't despair if you don't have a streaming service with lots of user interaction to collect data from; chances are that there's still plenty to collect.

Ideally, a recommender system collects all data about a user when they interact with the content—down to measuring brain activity, adrenalin released in the blood when exposed to an item, or how sweaty the user's hands get. The more we live our lives connected, the more realistic this scenario might sound.

In the movie *WALL-E*, humans have devolved into shapeless things who live all their lives in a chair in front of a screen, where everything about them is fed into a computer. (Come to think of it, I spend most of my days sitting in front of a screen. But at least I move between screens.) Because most people have other things to do besides being hooked up to a recommender system, we need to lower expectations a bit. But with the web, we've become closer to users than any physical store ever could, so it's possible to learn many things.

Content affiliation to provider

In chapter 1, one of the dimensions in the taxonomy was purpose. Purpose is important because it might result in particular strategies for calculating suggestions, as well as what you want to suggest.

Take, for example, a film: if you watch a bad film on Netflix, it tells you something about the quality of the content on Netflix and, therefore, says something bad about Netflix. If Amazon sells a Blu-ray disc of a bad film, you probably wouldn't think any less of Amazon, but if you were looking for the film and couldn't find it, that would make you think less of Amazon. I mentioned this in section 2.1 with the T-shirt analogy, but a good point is worth making twice.

The purpose of Netflix is to show good films that you like. Amazon shows you things to buy; whether you like them isn't so important. Amazon spends many resources on asking customers to write reviews and rate content, so it might not be completely fair to say it doesn't care, but for the sake of example, it will do.

2.2.1 Capturing visitor impressions

To better illustrate the events that occur in the lifetime of a consumer/product relationship, I've divided it into the following steps, illustrated in figure 2.3:

1. Consumer browses. As in a physical shop, the consumer looks around to see what's there, with no specific goal. What's noteworthy is where the consumer pauses and shows interest.
2. Consumer becomes interested in one or more products. It might be that the consumer knew from the start that they were looking for something specific or it might be by chance.
3. Consumer adds product to basket or a list with the intent to buy.
4. Consumer buys products.
5. Consumer consumes product. For example, the consumer watches the film or the consumer reads the book. If it's a trip, the consumer goes on the trip.
6. Consumer rates the product. Sometimes consumers return to the shop/site to rate the product.
7. Consumer resells product or otherwise disposes of it. The consumer lifetime of the product is finished; it's disposed of, deleted, or resold; in which case, the product probably goes through the same cycle again.

We'll look at what can be collected at each of these steps a bit later. But note that explicit feedback in the shape of a rating is done in step 6 or later. That's late in the process. Therefore, even if ratings are always the first thing people talk about, you should record data prior to that.

2.2.2 What you can learn from a shop browser

Now to go into detail about what's happening in steps 1–3 in figure 2.3. A *browser* is a customer who looks through content. They might randomly go through many different things but often pause at content that seems relevant or interesting. In a physical shop, a browser strolls through the store, not showing any direction or purpose. In a sense, the customer is gathering intelligence for later buys.

> **A browser**
>
> A *browser* is a customer looking through content. A browser as I said earlier, should be exposed to as many different things as possible, and suggestions should reflect that. If you could classify that a visitor is a browser, you could use that information to produce suggestions that fit that mood.

What you need to collect here is where the browser stops and investigates. It's also worth keeping track of what the browser sees without showing any interest. But can you be sure that a page view (product view) is always good?

PAGE VIEW

A *page view* in an e-commerce site can mean many things. It can be an indication that the visitor (or browser) is interested, but it could also identify someone who's lost or clicking randomly. In the latter instance, more clicks aren't positive. A lost user shows up as a visit with many clicks but no conversions.

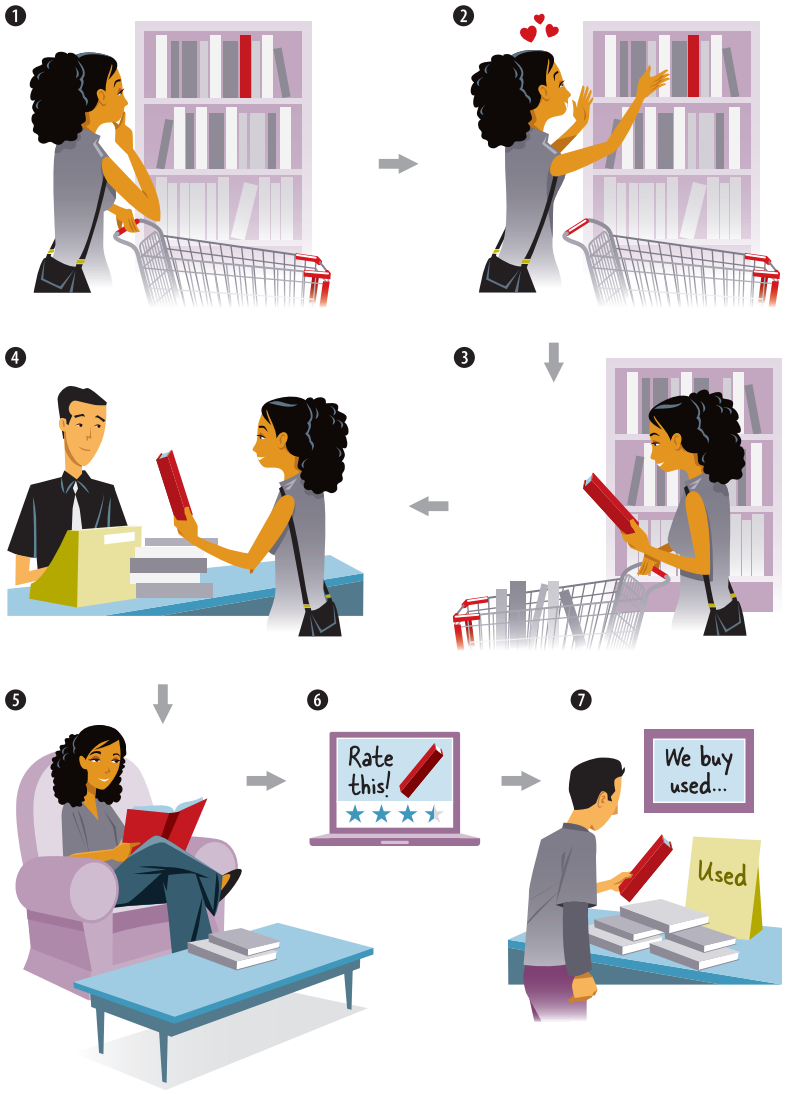

Figure 2.3 Consumer/product relationship lifecycle

On the other hand, a great recommender results in fewer page views. That's because people will find everything they're looking for from the recommended links and products without needing to browse around first.

PAGE DURATION

To determine what a visitor browsing your site is interested in, you can measure the customer's duration on a content page. But is that straightforward? It is if you assume that the customer isn't doing anything else and that the next thing the customer does

is to go to a new page by following a link on the current page. Table 2.3 shows one way to interpret the possible meaning of how much time a browser spends on a page.

Table 2.3 Page durations and a possible interpretation

Duration on page	What it means
Less than 5 secs	No interest
More than 5 secs	Interested
More than 1 min	Very interested
More than 5 mins	Probably went to get coffee
More than 10 mins	Interrupted or went away from the page without following a link

Adjust the duration times to fit your domain, but I think most would agree that these interpretations could be true. Which of these is worth saving? Well, all of them. Less than 5 secs is a dislike, 5 secs to 1 min could mean "interested," 1–5 mins could mean the user thinks "this is great," and 5 mins and more is hard to say. All of these depend on the content of the page. It's not an exact science.

EXPANSION CLICKS

Besides page duration, there are other ways to record user interest in the content. Add small control interactions that help you determine what the user is doing. For example, websites often use links to more information, as shown in figure 2.4. This is convenient for the customer, who can get a quick overview or expand the link if

Figure 2.4 When an expansion link is clicked, it's an indication of the visitor's interest. This is an example from Amazon.co.uk.

they're interested. Similarly, a user might scroll down to see reviews or technical details. If a user does one of these things, consider it a sign of interest.

Social media links

You can also add social media buttons (figure 2.5) for people who like something so much they want to share it with others. You can't control what happens on Facebook or Twitter, or one of the other social media sites, but you can collect the event of a consumer sharing something.

Figure 2.5 The usual gang of social media links

Save for later

A Save for Later feature that lets users add things to lists is powerful. If a customer finds something of interest, it's a good idea to provide the user the capability to save it for later (if they don't buy it immediately). This can be as simple as having a link for bookmarking the page. Even better, have a wish list, favorites list, or watch list, depending on the type of content. Other signs of interest could be downloading a brochure, watching a video about particular content, or signing up for a newsletter on a specific topic.

Search terms

Visiting a website can mean that people are either browsing or that they're looking for something in particular. If the page is laid out well, most customers find what they want quickly. Netflix says that every time someone starts searching, it's seen as a failure of the recommender system because it means people didn't find anything they wanted to watch among the recommendations. I'm not sure I would agree with that because often I use the search function because someone has recommended something that might be outside of what I usually watch. In any case, a search term is one of the best ways of understanding what a customer is looking for.

Figure 2.6 shows a Netflix search window. The site has more movies than are available to watch, so if you search for `wonder woman`, it shows you similar titles, even though *Wonder Woman* isn't part of the catalog.

Even if the system can't provide the content searched for, registering the event is worthwhile. If a user looks for a film, you know they're interested in something about that content. With this knowledge, your recommender can suggest something similar.

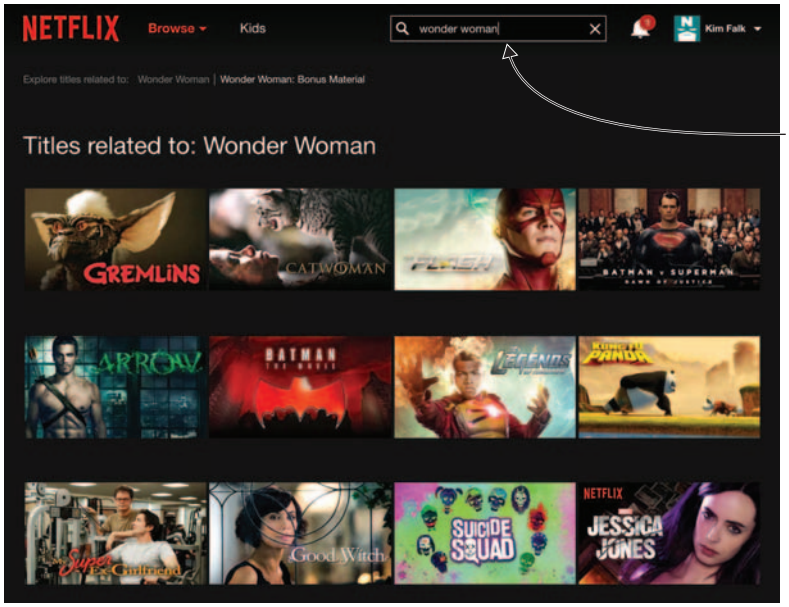

Figure 2.6 Netflix search result window, searching on wonder woman

Linking searched items with the resulting consumption

Another thing to consider about *search terms* (what a customer types in the search field) is that it's a good idea to connect what's searched for with what's consumed. Say a user searches for *Star Wars* and looks at *Harlock: Space Pirate*, which has a reference to *Babylon A.D.*, and the user ends up watching that. Maybe it's worth putting *Babylon A.D.* in the search result for *Star Wars*.

2.2.3 Act of buying

Buying something means that the consumer considers the item useful or likeable—or maybe it's a gift. It's not easy to determine whether a purchase is for the buyer and thereby another piece of evidence that can be used to understand the user's taste, or if it's a present and something that should be disregarded.

Figuring out which purchases are presents and which aren't is an interesting problem. An item that's different in taste from the items consumed by the user so far could either be an indication of a new dimension in the user's taste or a present. Either way, it's seen as an *outlier* in the data.

Graphically, an outlier shows up as a point far away from the main body of points, as shown in figure 2.7. Because you can't be sure about what they indicate (present or

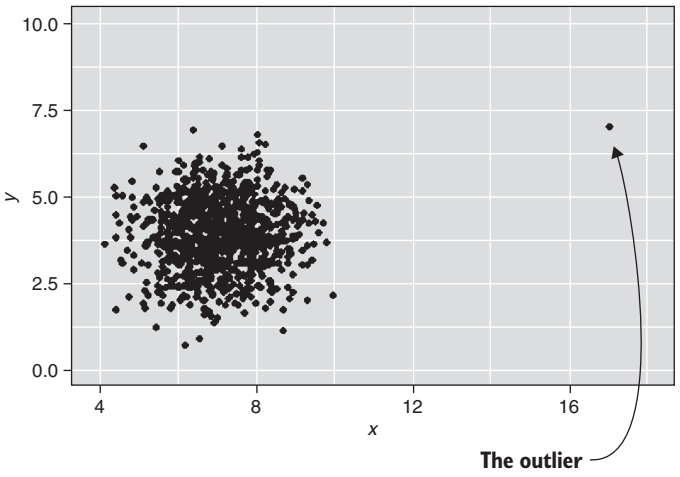

Figure 2.7
Example of an outlier

a new interest), it's often better to disregard outliers. Conversely, it could also be a first indicator of a new trend, which is an opportunity that could be explored.

The act of buying something means that the product was presented in a good way, although it doesn't say anything about whether the consumer likes the product. At least, this is true if it's the first time the consumer buys the product. People might argue that every time a consumer buys a banana, that indicates more stars on the rating for bananas. A first buy might not be much of an indication of approval, but a second buy is. Either way, a buy can be regarded as something positive.

2.2.4 Consuming products

When something is bought, the shop loses contact with the product and the ability to track how it's used—if it's not a streamed product or service provided from the website.

ENDOMONDO

Movies and music aren't the only content that's consumed online. Endomondo (www.endomondo.com) is another example of a site providing online services. The social fitness sports network allows users to collect statistics about their activity using a sports tracker.

Endomondo keeps track of how much you use its features, a basis for the company to recommend similar services or to understand where they should develop new ones. Telephone companies also measure how consumers use their phones; they can track us in scary ways. The following section discusses what you can learn from streamed products.

STREAMED PRODUCTS

In the case of streaming services, music, film, or even books, all user interactions can be considered an implicit rating. Listening to a song is an indication that the user

likes it. But this data can be analyzed even more. The following list shows user interactions with music or movies:

1. *Start playing*—The user is interested; that's already positive.
2. *Stop playing*—Oh wait, maybe the user was curious enough to start playing but thought it was so bad that they stopped. Stopping a song within the first 20 secs (or a film within the first 20 mins) can be a bad sign. Stopping close to the end can be considered something else.
3. *Resume playing*—Okay, forget about all the negative implicit ratings the system registered. Resuming something after stopping can mean various things. If playing is resumed within 5 mins, somebody or something probably interrupted the consumer, so the stop and resume shouldn't be counted. But if the consumer stops and then resumes 24 hrs later, the consumer probably likes the content.
4. *Speeding*—If the user skips something in the middle, it's probably not a good sign—but only if it's the first consumption. If it's the tenth time a film has been watched, for example, skipping a boring scene probably doesn't make the overall perception of the film worse. The technical proofreader of this book noted that with music, he often skips through songs to understand a song, or whether it's worth listening to. Songs have less context, so with skipping, you can get a sense of whether you'll like the content or not. This wouldn't work with movies.
5. *Playing it to the end*—We have a winner! This is a good sign—it might not be something that the user would rate high, but if they sat through the whole thing, it probably means that they'd watch similar films. (Played to the end means played until the film ends and the credits start rolling.)
6. *Replaying*—Replaying content might mean something good for film and music, but for a site offering educational videos, it can also mean that the topic was too difficult.

These steps work on most streaming products. How to collect evidence from streamed products depends on the type of player that's used.

In the case of Endomondo, which is also a kind of streaming service, these steps and explanations don't really hold. In this sense, if you start Endomondo and indicate that you started running (by pressing the Play button), it probably doesn't have anything to do with how much you like the app if you pause it after 10 k; that may mean that you should get in better shape.

2.2.5 Visitor ratings

Finally, you've arrived at what everybody talks about—the ratings. Netflix has a motto:

The more you rate, the better your suggestions.

That might be a truth requiring modifications, as you'll see later. Most recommender systems use ratings, but those user ratings are usually weighted against user behavior. These systems use ratings only as a starting point. What you want is to capture the behavior of users.

Many sites enable users to review content that they've viewed, bought, or used. This enables the system to gain a better understanding of what users like and thereby what to suggest in the future. Adding a certain number of stars (or hats, smileys, or something else) determines a rating, but behind the scenes that's only a number on a scale. Amazon, as with most places, tries to help you with what each number of stars means by providing tooltips. Figure 2.8 shows an example of a book review on Amazon.

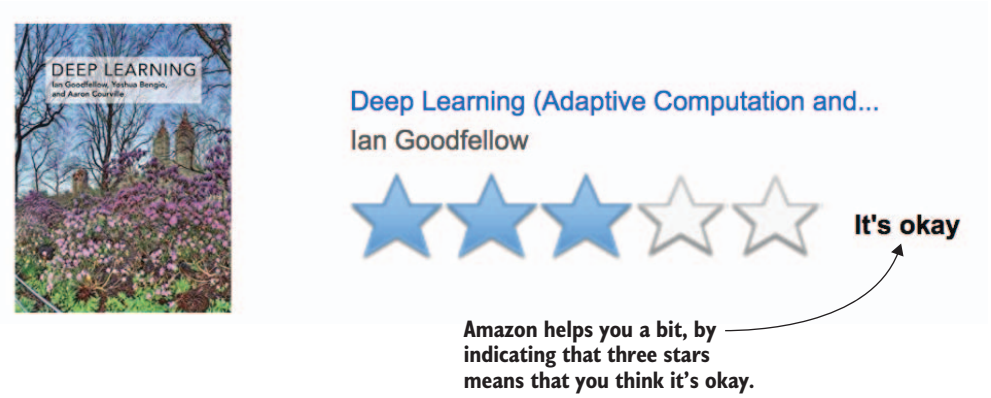

Figure 2.8 When rating something on Amazon, it provides hints as to what the number of stars means.

On Amazon, as the user passes the mouse over the stars, a description is shown. In this case, four stars means in effect, "I like it." In addition to ratings, certain sites, such as TripAdvisor, encourage users to write a review.

You could say that a five-star rating plus a written review counts as more than a five-star rating alone, because the person who writes the review puts more thought into it. The same is true for one-star ratings. But if a person accompanies all ratings with a written review, then it doesn't mean more. Could buying something and not rating it indicate something about your preference?

A SENSE OF CONTROL

When just-add-water cake mixes (figure 2.9) first arrived in stores, they were a huge failure. The product seemed perfect for busy consumers: the only thing they had to do was to add water. Through consumer studies, it was discovered that the problem wasn't that the process was easy, but that it was *too* easy. Baking a cake is about creating, but pre-preparing everything made the process too easy; it took away the consumer's sense of control. The manufacturers said, "All right, we'll let them add eggs also." It was less expensive to produce and the consumer felt empowered. When the cake mixes were introduced again, with instructions to add water and eggs, they were a huge success.

Many sites let users add their preferences for the very same reasons: to give the consumer a sense of control over what the system believes is that consumer's taste. Netflix states that many people indicate that they like documentaries and foreign

Figure 2.9 Chocolate cake mix. When introduced, the consumer needed to add only water. But it wasn't a success until consumers were instructed to add water and eggs.

movies but watch American sitcoms. Then what should Netflix suggest: things that make you feel bad about your choice of entertainment or things that you want to watch?

This is one of the reasons it's hard to use data sets with ratings to test whether a recommender system is good. Data sets can test whether your prediction calculations work, but not whether the system will attract more users.

SAVING A RATING

When a user adds a rating, it's an event, and that event should still be saved among the evidence as any other event. It might also be worth saving directly in your content database, so you can show average ratings when you present the content to the user.

NEGATIVE USER RATINGS

Things get a bit tricky if you as a consumer want to indicate that you dislike certain content because to review something you have to give it at least one star. Zero stars means no rating at all. If you hate something, you don't want to give it any stars, not even one. In a sense, not rating it is better than "I hate it," which you can indicate on Amazon with one star as shown in figure 2.10.

Not liking something can mean not bothering to rate it. But if something really irritates you, you might want to release your frustration somewhere, and that's often in the form of a negative review.

The rating in figure 2.10 is not mine. I definitely recommend *Deep Learning* (The MIT Press, 2016) if you have a background in machine learning. Otherwise, you might be better off starting with *Grokking Deep Learning* by Andrew Trask, (Manning, 2016).

Figure 2.10 At Amazon you show you hate something with one star. Not that I dislike this one. If you want to get into *Deep Learning*, then this is the book to read.

VOTING

Many sites have had success creating a community around users voting whether something is good or not. For example, TripAdvisor's only service is the rating of hotels and restaurants. Another such example is Hacker News (https://news.ycombinator.com/), where the users are responsible for adding content, which can be links to articles and blog posts about "anything that good hackers would find interesting." When something is added, you can *up-vote* it. The more up-votes it gets, the higher it stands on the page (it's almost as simple as that; you'll take a better look at the algorithm later). Sites that use voting are called *reputation systems*.

2.2.6 Getting to know your customers the (old) Netflix way

With a page that's almost completely built based on suggestions, it's important to gather as much input as possible about a consumer's tastes. If Netflix thinks that your tastes are different, finding what you want to watch can be difficult. This is another point for people who claim that recommender systems are manipulative. Because the system doesn't provide equal opportunity for all content, it therefore manipulates you. I understand the argument, but I disagree.

Netflix once offered the user the capability to assist in creating a taste profile.[3] This feature is no longer available, but when it was, you could find it in the Taste Profile menu shown in figure 2.11. It let the user rate shows and movies and select genres by indicating how often the user felt like watching, for example, *Adrenaline Rush*. Netflix could then check whether the user's ratings matched their current opinions.

Netflix used this manual input of a taste detail to offer better suggestions. Asking the user for help with the taste profile is a method often used to enable the system to give suggestions for new users.

[3] For more information, see https://help.netflix.com/en/node/10421.

Figure 2.11 Netflix Taste Preferences, 2015. The feature is no longer available.

2.3 Identifying users

Collecting data on users works only if you have a way of uniquely identifying customers. The best way to do that is to make the customer log in on your page so you've positive identification. Another alternative is to use cookies.

Usually sites start out by setting a cookie and connecting all information to that cookie. If the user then provides identification by logging in or by creating a profile, all information from the cookie is transferred to that account. Be careful with cookies because the computer could be a public or a family computer used by several people. Saving data to the cookie ID over several sessions can be misleading.

If you don't have logged-in users, there's also a cross-device problem, which means that even if you recognize a user on one device, your system can't recognize the same user across different devices. Certain services can help you with this but always with uncertainty. Try to make users register and log in when possible. It goes without saying that personalized recommendations can only work if you recognize the user.

2.4 Getting visitor data from other sources

Your site is unique, and the data that can be collected on your site is the data that's best suited to reveal the behavior of your customers. But what if you could cheat a bit and get data from somewhere else?

Social media is a good place to start, and, if you're lucky, a visitor on your site has liked something that matches the content in your catalog. Depending on your content, the social media site could be Facebook, LinkedIn, or similar sites. Many will think that, yes, connecting to Facebook is good, but if you aren't dealing with films or books, what's the purpose of adding data from Facebook? Most sites, however, will benefit from getting something as simple as the age of a visitor or where they live.

There might be other ways to gain knowledge too. A recommender of pension plans could benefit from knowing whether a customer reads finance books, for example. That would be an indication that the customer is interested in the stock market and will probably be more interested in pension plans enabling the customer to have more control over a portfolio.

Another thing to consider is that many algorithms calculate recommendations based on similar users. If the system has the same data about many users, even if that

data might not be relevant for the content of your site, it still enables the system to find similar users that can then be used to create recommendations. I'm sure a car dealership site can find that there's a special kind of film that most SUV owners like, or that a makeup site can recommend something based on age and gender. People who work in IT probably like gadgets instead of phones, and train drivers might like sunglasses.

2.5 The collector

We're now going to look at an evidence collector implemented for the MovieGEEKs website. We'll look at the essential parts and then leave it to the interested reader to explore the details within the code.

Because of performance and reliability of your website, it's better not to add evidence collection to your current (web) application. Instead, add it to a parallel structure that supports what you want to achieve. This lets you move the evidence collector to another server if the users go wild after adding your recommender system or the load on your site gets close to its limit—for scalability.

This evidence collector has two logical parts:

- *The server side*—In our example, the server side is built using a Django web API that works as an endpoint and can be used by anything that a user is in contact with. Mostly this means a web page, but it could also be an app on a phone or any kind of device connected to the internet—anything that collects relevant events. When the server receives a notification that an event has occurred, its only job is to save it. A web API is an HTTP address configured to receive this kind of message and can be implemented essentially by any type of framework.
- *The client side*—There won't be a client part in the traditional sense because there're no web pages that can be requested. The client side consists of a simple JavaScript function that posts evidence to the evidence collector on the server.

With a collector in place, you'll be prepared to start collecting data either on MovieGEEKs site or on your own. To relate the evidence collector to the rest of MovieGEEKs architecture, figure 2.12 highlights elements of the evidence collector in the architectural diagram from chapter 1.

You might be thinking, why not use Django logging to save all the hassle of adding another app? But consider this:

- Django logging works on server-side code execution, so you have no way of saving user behavior (except by putting it in a kind of session state and saving it in the next request). All the events such as hover and scroll would be lost.
- The collector lets you receive evidence from everywhere, not just a website. The future is around the corner, where evidence is something that you record from phone apps or other strange gadgets that are coming out, as well as from physical shops.

Because the data collected is simple, it's often best to save it in a comma-separated (CSV) file. In this way, it's easier to move it around if the system that uses it is somewhere

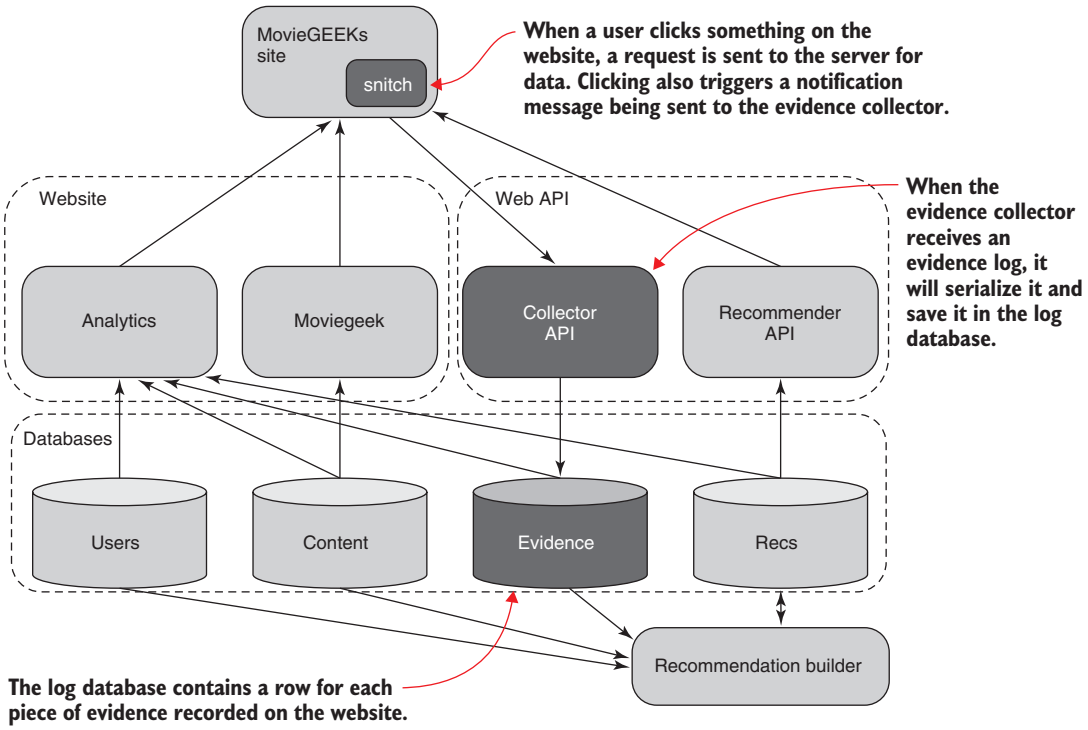

Figure 2.12 MovieGEEKs' evidence collector and logger architecture

else. The CSV file can then be fed to a service that ingests the data at a speed the system CSV can handle. That way, you always have a buffer to ensure that you aren't overloading your system. Because a CSV file isn't easy to query, and while learning about recommender systems, it's important to query the data. Instead, you'll use a database.

2.5.1 Building the project files

Getting the site up and running is fairly easy. You need to download or clone the GitHub repository at http://mng.bz/AU9X. After the download completes, follow the instructions in the readme.md file.

2.5.2 The data model

It's important to collect most kinds of interactions. The previous section indicated that all you need are three things: user ID, content ID, and event type. You really need a few more things to make it work, as listed in the data model shown in figure 2.13.

The `session_id` uniquely identifies each session. Later, you can add more to this model such as device type or context. But for now, you can start with this.

Logger	
date:	datetime
user_id:	varchar(64)
content_id:	varchar(16)
event:	varchar(200)
session_id:	integer

Figure 2.13 Data model of an evidence logger

You'll find plumbing around the logger, which you're welcome to look into, but it's a web API that's open for one type of request containing the data as shown in figure 2.13. You'll virtually tag along on a request a bit later on, when we talk about how it's hacked into the MovieGEEKs website. But first, let's look at what you should do on the client side.

> **A note on time**
>
> Recording time is always a bit tricky because you must take into account different time zones. Using all local time zones means that you can have problems with the order of events, because events happening at the same time in different time zones will be farther apart. But recording local times means that you can work with phrases such as "in the afternoon" on a global basis, rather than having to look at different time intervals for each time zone.

2.5.3 The snitch: Client-side evidence collector

Evidence can be collected as events from anything that interacts with the user, from a phone app to a device you put in your shoe when jogging.

The event logger is a simple JavaScript function that calls the event collector. It doesn't do anything if there's an error, because this isn't something end users can do anything about. In a production environment, it's worth keeping track of whether evidence is being recorded, but this probably isn't the right place to do it.

The snitch should be in the project where you want to collect data. You'll find a file called collector.js in the /moviegeek/static/js folder containing the function shown in listing 2.1, which creates an AJAX call to the collector.

Listing 2.1 Creating an AJAX call to the collector: /moviegeek/static/js/collector.js

```
function add_impression(user_id, event_type, content_id,
                        session_id, csrf_token)
{
    $.ajax(
            type: 'POST',
            url: '/collect/log/',
            data: {
                "csrfmiddlewaretoken": csrf_token,
                "event_type": event_type,
                "user_id": user_id,
                "content_id": content_id,
                "session_id": session_id
               },
            fail: function(){
                  console.log('log failed(' + event_type + ')')
                  }
     })
};
```

Makes an AJAX call

To be RESTful, the message sent is a POST because you're adding something to the db.

A CSRF middleware token that allows your site to call a site from a different domain.

Shows the three important data elements

Shows a unique session ID

If it fails, then write something out to the browsers debug console. Don't show the user anything.

The calls from the function in listing 2.1 will eventually be received by the log method in /moviegeek/collector/view.py shown in the next listing.

Listing 2.2 Receiving calls from listing 2.1: /moviegeek/collector/view.py

```
@ensure_csrf_cookie
def log(request):

    if request.method == 'POST':
        date = request.GET.get('date', datetime.datetime.now())

        user_id = request.POST['user_id']
        content_id = request.POST['content_id']
        event = request.POST['event_type']
        session_id = request.POST['session_id']

        l = Log(
            created=date,
            user_id=user_id,
            content_id=str(content_id),
            event=event,
            session_id=str(session_id))
        l.save()
    else:
        HttpResponse('log only works with POST')

    return HttpResponse('ok')
```

This method is only interested in POST type messages.

Creates a timestamp to add to the created field

Saves a log entry in the database

Responds nicely even if it isn't a POST message

2.5.4 Integrating the collector into MovieGEEKs

The MovieGEEKs app covers the examples shown in table 2.4. The examples are similar to the ones listed in table 2.1, which depict a use case on the Netflix site. I've added the table again here with a new column showing the event data that will be collected.

Table 2.4 MovieGEEKs evidence points

Event	Meaning	Evidence
Clicking a genre such as Drama	User is interested in the theme (here, Dramas).	`(Kimfalk, drama, genreView)`
Placing the mouse over a film such as Toy Story (requests an overview of the content)	User is interested in the movie.	`(Kimfalk, ToyStory, details)`
Clicking the film (requests details of the film's content)	User is further interested in the movie.	`(Kimfalk, ToyStory, moreDetails)`
Clicking Save for Later	User intends to watch the movie.	`(Kimfalk, ToyStory, addToList)`
Clicking the Buy link	User watches the movie.	`(Kimfalk,ToyStory, playStart)`

LOGGING GENRE EVENTS

The following events have been implemented in the templates/moviegeek/base.html file. The first event to log is the user clicking a genre. The genres are listed on the left side of the screen as shown in figure 2.14.

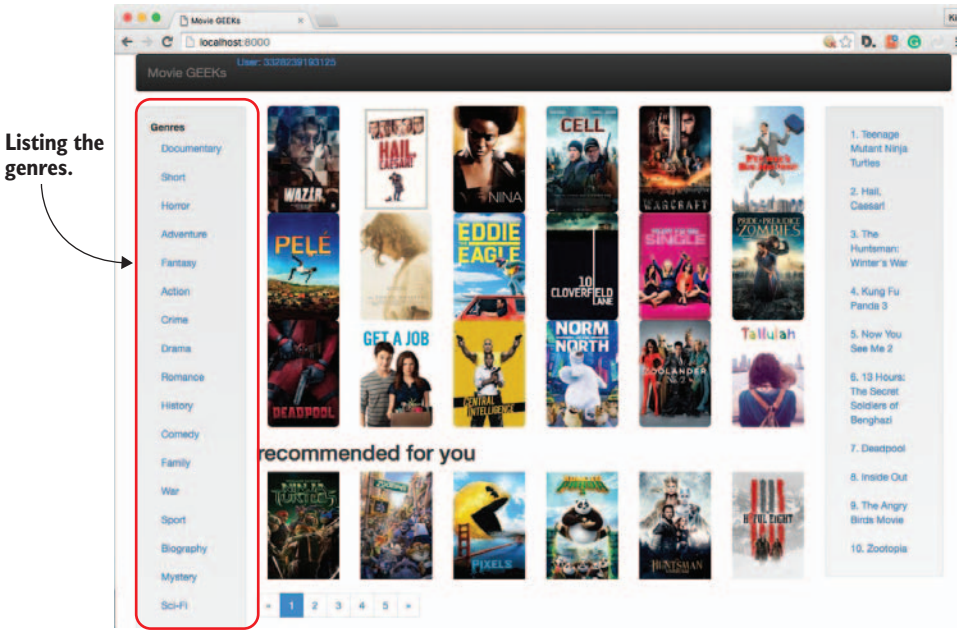

Figure 2.14 MovieGEEKs homepage

To register clicks on a genre, an `onclick` attribute is added to each link. When the active user clicks a genre, it fires an HTTP `POST` request to the collector. We won't go too much into the code because it's quite repetitive, but look at figure 2.15 to understand what happens when a user clicks a genre.

1. User clicks a genre.
2. The `onclick` event executes, which calls the JavaScript function in listing 2.1.
3. The `add_impression` function executes.
4. The HTTP request is received by the web server, which delegates it to the MovieGEEKs site.
5. The MovieGEEKs site does a lookup in the URL list and delegates everything that has the URL /collector/ to the collector app.
6. The collector app matches the log/ to a view method.
7. The view method creates a log object.
8. Using the Django ORM system, writes the log object to the database.[4]

[4] For more information, see https://docs.djangoproject.com/en/1.9/topics/db/.

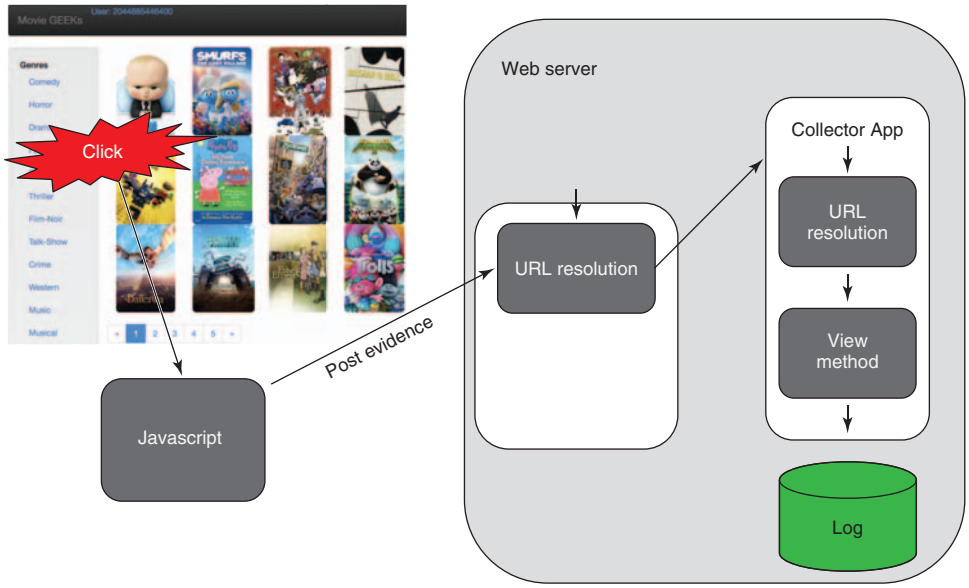

Figure 2.15 What happens when a user clicks a genre

LOGGING POPOVER EVENTS

When a user clicks a film, a popover appears. A *popover* is a fancy name for a big tooltip. Figure 2.16 shows what one looks like. Because the user clicks, this indicates that the user might be interested in the film and is, therefore, something you should log. This is done by adding an event handler that calls the collector every time a popover is shown.

LOGGING MORE DETAILS EVENTS

A user who finds the information in the popover box interesting can click the More Details link. This is also something to note because it shows further interest in the movie.

LOGGING SAVE FOR LATER

Instead of clicking More Details, the customer can add an item to a list. That's an important event because it indicates that the user is planning to buy or consume it later. This functionality is good to have. It's a link on the details view, which records an event that you'll call `saveforlater`. You can also record other events, but these suffice for the purpose of this example.

2.6 What users in the system are and how to model them

Before moving on, you also need to think a bit about users. We've talked about the behavior of users, but what other things might be useful to consider when representing what users know and care about. As mentioned earlier, when it comes to knowing

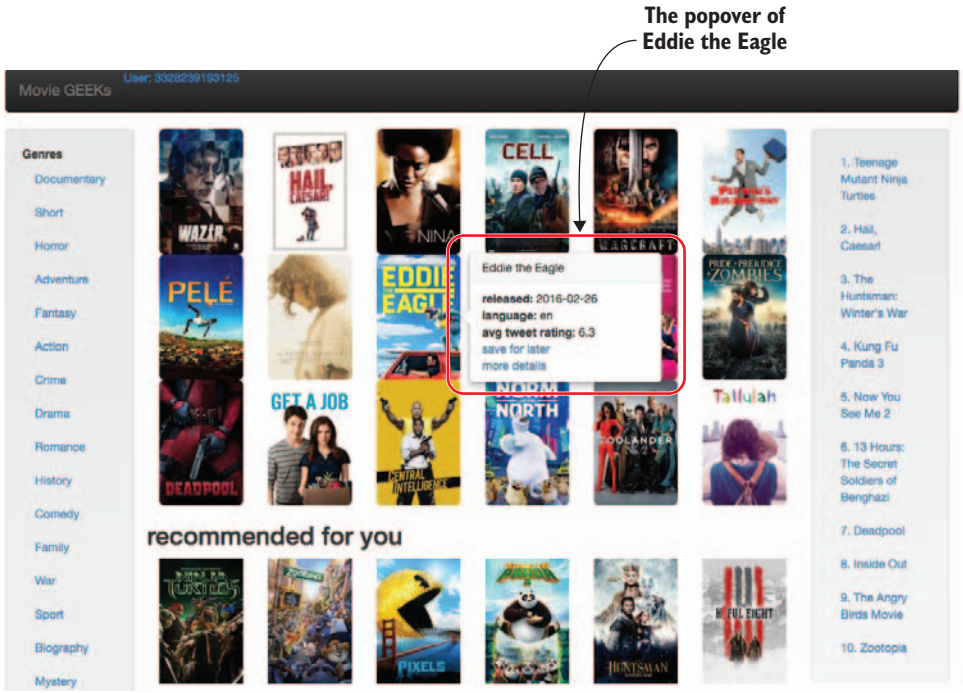

Figure 2.16 MovieGEEKs front page with popover

consumers, many other things can be relevant. You need a user model that you can translate to a database table and use in addition to the evidence.

What could be relevant to know about a user? Again, the answer is, as always, *it depends*. I hate that answer because it's one of the most useless replies, up there with "yes, but no" and "it was once true, but not now." Assuming you're of the same conviction, let's pretend that answer doesn't exist and look at different scenarios where you can say something about what would be relevant. Another answer could be "everything"; in theory, everything could be pertinent to recommending products.

If you're implementing a recommender system on a website such as JobSite (www.jobsite.co.uk), then it's relevant to gather information such as current position, education, years of experience, and so on. If you're looking at a pension site, you probably need the same things as JobSite, but also health data (such as how often you've been to the hospital and what medicine you're taking) will also be of interest. Book sites could also use all the things mentioned previously, because these are all things that say something about books you might be interested in reading. But most book sites probably will use things such as taste and buying habits.

If you shouldn't say "it depends," then let's say "everything" and take it from there. Let's say that Pietro is created in your system (see figure 2.17). What information would be good to store on Pietro? If you had the possibility to retrieve the information

in figure 2.17, what should you save in your database? In this day and age where storage is so economical, why not save all of it. You'll see what you can do with it in future chapters.

Figure 2.17 A new user called Pietro

Ideally, you want to keep a list of key-value pairs next to the user identifiers such as user ID, email address, and possibly other additional information. Again, remember you're doing a recommender system, so typically you'd also save a shipping address, and things like that. But for the purpose of the MovieGEEKs site, that isn't so important. This information gives you a data model like the one shown in figure 2.18.

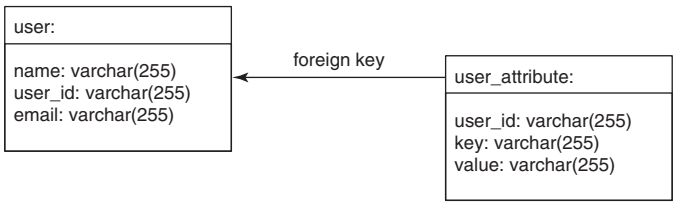

Figure 2.18 Generic data model

There are a couple of things to aim for. You want

- Flexibility to save everything
- Simplicity to make code readable

But these are pointing in opposite directions. You should, therefore, do a less flexible, but easier-to-use implementation (somewhere in the middle ground of what we discussed earlier). You can create a table containing most of the previous attributes, plus one that contains any extras you might need later. Your data model for a user might look like one shown in figure 2.19.

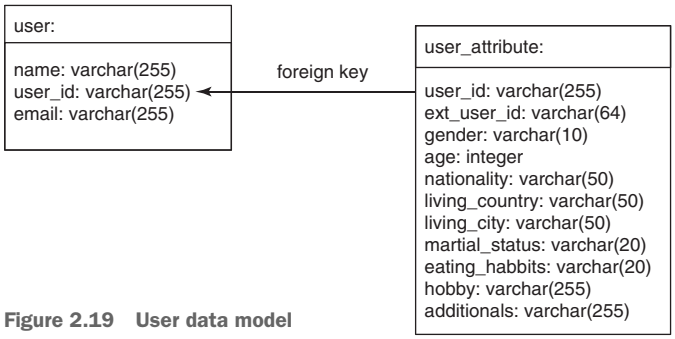

Figure 2.19 User data model

Continuing with our user Pietro, you'll save the data shown in figure 2.20. This data model isn't too flexible, but until you need a flexible data model, it's better to keep the level of complexity as low as possible.

If you have regular visitors, your log might already have something to show you about your visitors, and that's great. Nevertheless, it's always good to have user information to check how your algorithms work. Chapter 3 introduces personas and shows how to autogenerate evidence for them.

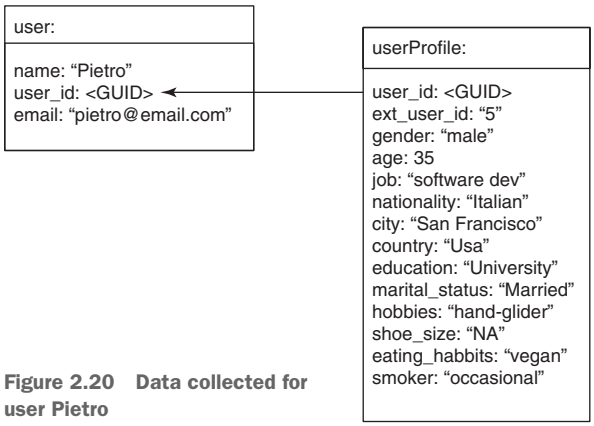

Figure 2.20 Data collected for user Pietro

Summary

- Log user behavior using a web API. This would possibly run in a different web application than that of the site to ensure that it won't cause the site to suffer in performance if the user triggers many events.
- Connect a snitch to a website by attaching a call to all events happing on a site.
- Good evidence provides information to the system about a user's taste. It's good to record all events because they might turn out to be useful later.
- Implicit ratings are deduced from the events triggered by the user, while explicit ratings are the actual ratings a user inserts.
- Implicit ratings are usually more reliable but only if you understand what each event means.
- Explicit ratings aren't always reliable because they can be biased due to social influences.