Overview of Recommendations Systems

Module 2

Recommendation Systems

Objectives

1. Compare the various types of recommendation systems.

Objectives

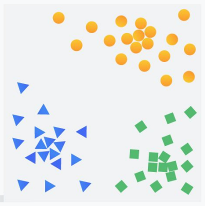
1. Compare the various types of recommendation systems.
2. Anticipate the problems that can arise when building a recommendation system.

Objectives

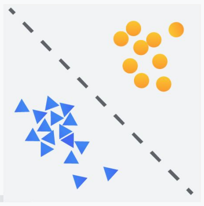
1. Compare the various types of recommendation systems.
2. Anticipate the problems that can arise when building a recommendation system.
3. Recognize common use cases for recommendation systems.

Machine learning types

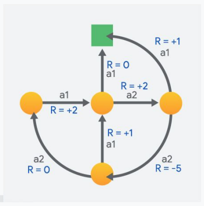
Unsupervised learning



Supervised learning



Reinforcement learning

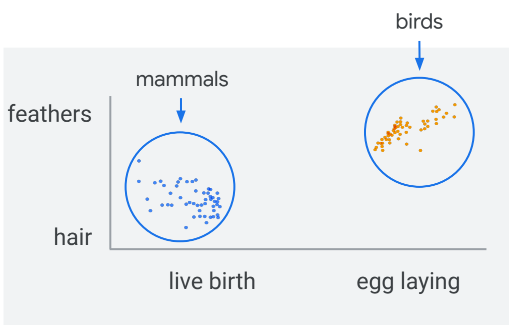


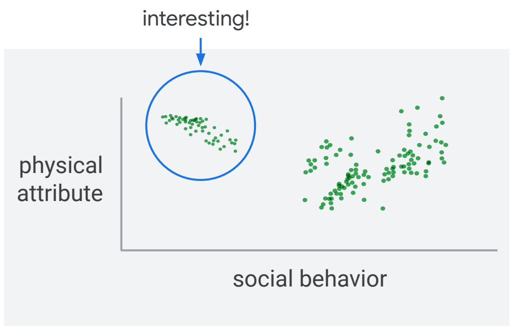
Title Safe >

< Action Safe

Unsupervised learning

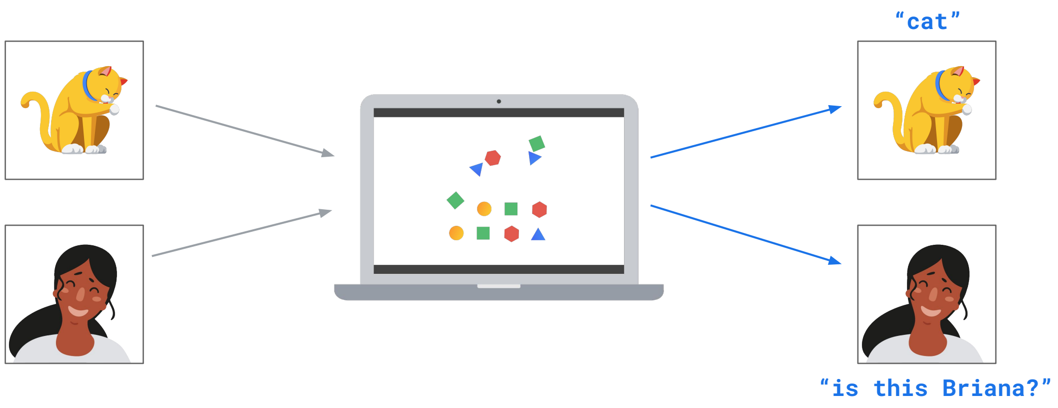
Find patterns or hidden structures in unlabeled data.





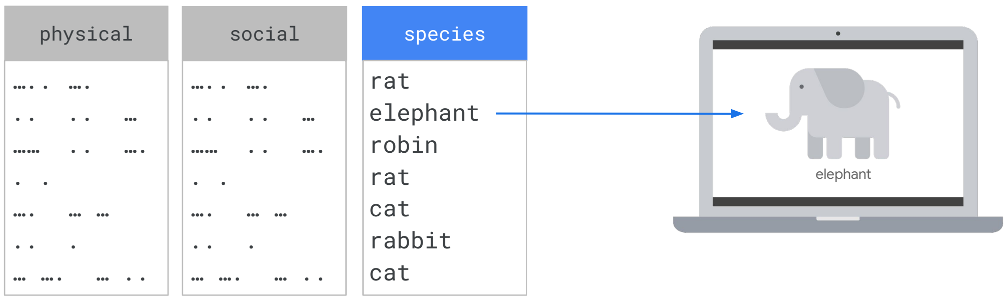
Supervised learning

Train a computer to apply a label to data.

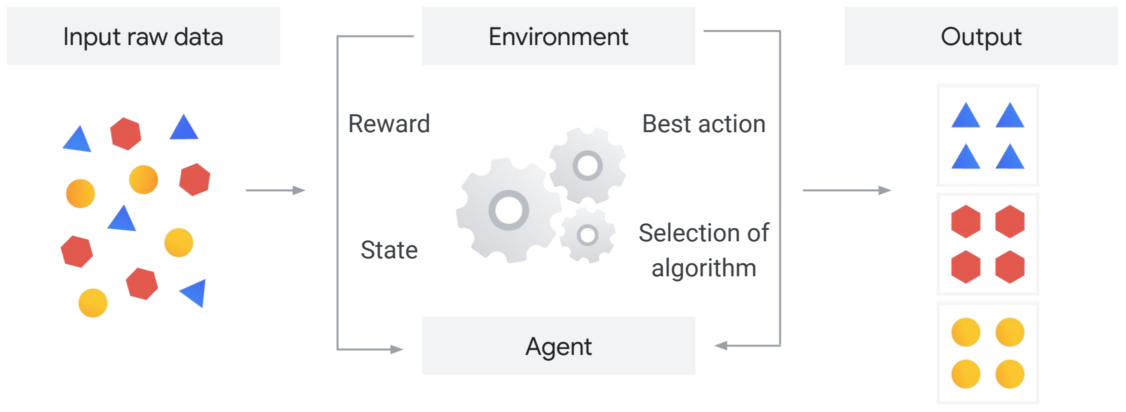


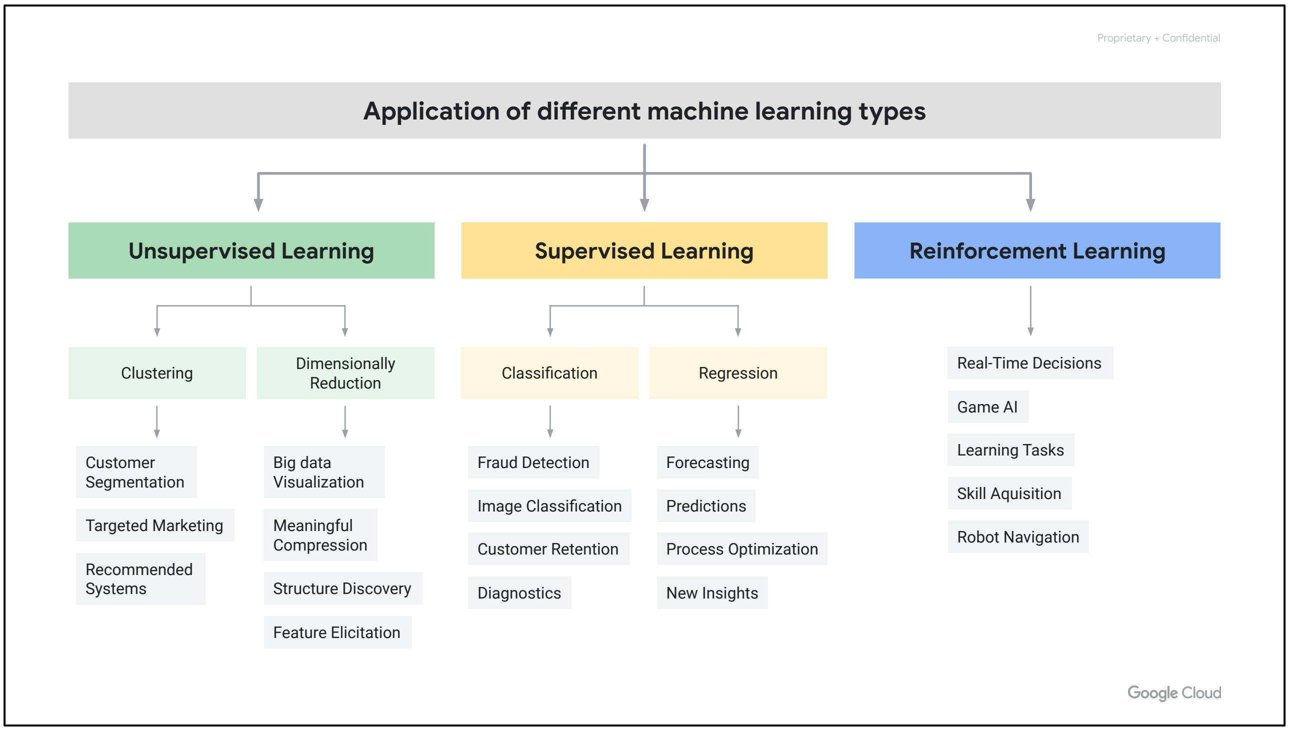
Supervised learning

Train a computer to apply a label to data.

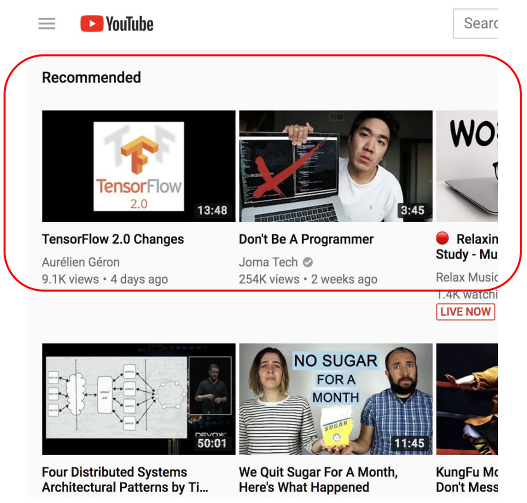


Reinforcement learning



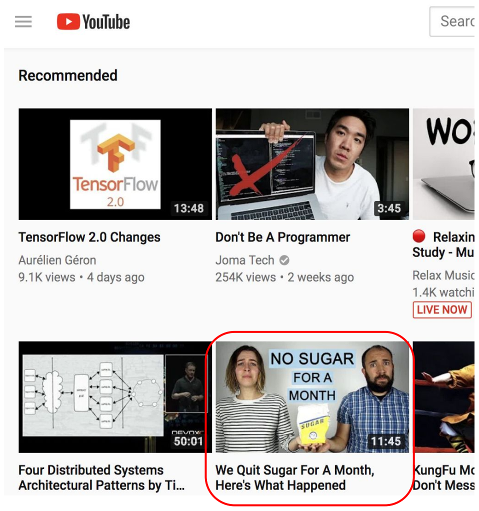


Application of different machine learning types



What do recommendations do? Well, a lot of things.

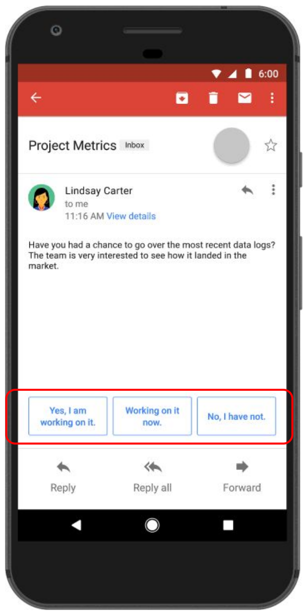
Recommendation engines identify things that a user may like, based on what they've watched in the past.
Like these video recommendations when you log in to YouTube. Most of the recommendations are about programming, Tensorflow, or machine learning.



...or they suggest new items that a user might not have thought to search for themselves, like this video on quitting sugar. It doesn't quite fit with the general theme of the other videos, yet it still might be relevant and interesting for this user.

This is especially useful for a product like YouTube, which has billions of videos in its catalog.
It's not feasible for a person to search through all that content.

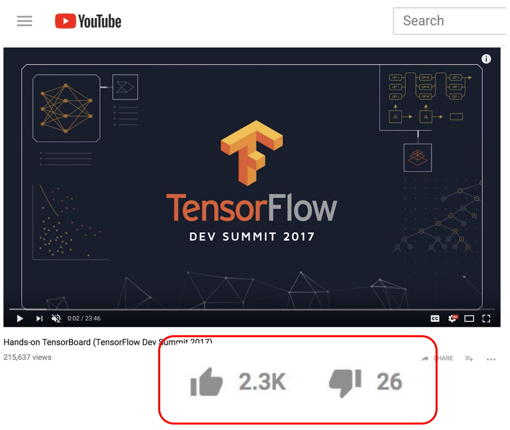
Recommendation engines provide a way of surfacing new content that a user might like, but didn't know to search for.



Lindsay Carter
to me
11:16 AM View details

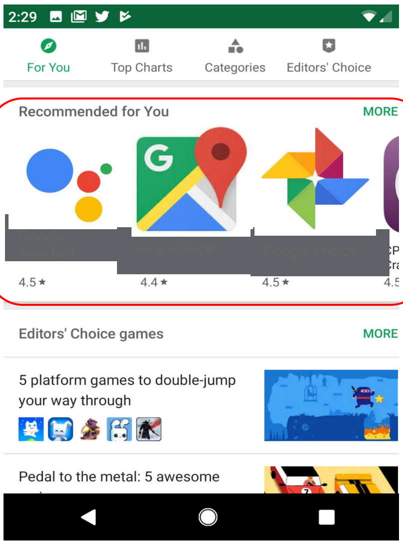
Have you had a chance to go over the most recent data logs? The team is very interested to see how it landed in the market.

Or consider the example of Smart Reply in Gmail. From the myriad possible responses to this email about project metrics, we are recommended three short replies.



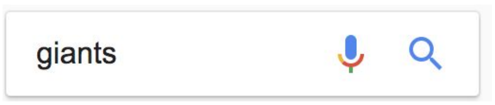
Recommender systems also allow us to personalize a user's experience.

By recording interactions such as "likes" and "dislikes," we can start to learn preferences unique to that user and thus personalize future content.....



and make personalized recommendations like this.

Recommenders can suggest new products or apps relevant to the interests that the user has shared.

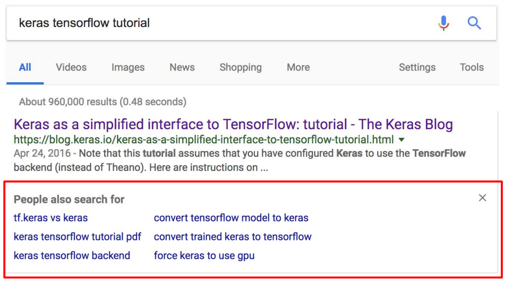


Google search is another great example of how recommendation engines can provide personalization. Your Google Search queries take into account your location, your user history, account preferences, and previous searches to ensure that what you are served is most relevant to you, the user.

For example, typing "giants" into the search bar might yield different results depending on where you're located.

If you're in New York, you'll might get a lot of results for the NY Giants football team.

However, the same search in San Francisco might return information about the San Francisco baseball team instead.


keras tensorflow tutorial

All Videos Images News Shopping More Settings Tools

About 960,000 results (0.48 seconds)

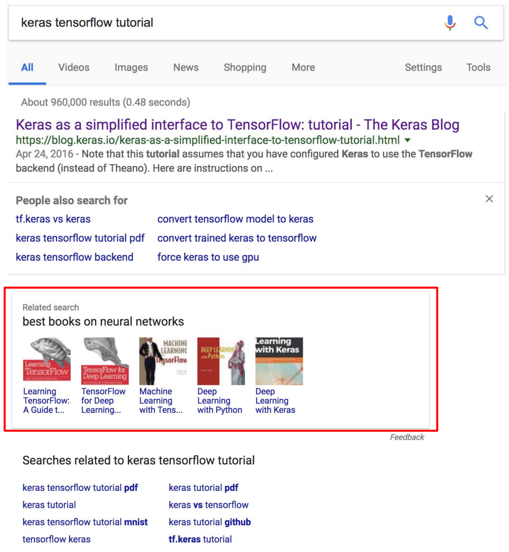
Keras as a simplified interface to TensorFlow: tutorial - The Keras Blog
https://blog.keras.io/keras-as-a-simplified-interface-to-tensorflow-tutorial.html
Apr 24, 2016 - Note that this **tutorial** assumes that you have configured **Keras** to use the **TensorFlow** backend (instead of Theano). Here are instructions on ...

People also search for

tf.keras vs keras	convert tensorflow model to keras
keras tensorflow tutorial pdf	convert trained keras to tensorflow
keras tensorflow backend	force keras to use gpu

Recommenders also help you find content that goes together—content you are already looking for or that you didn't know you wanted.

So, when you use Google Search to find "keras tensorflow tutorials," you are also recommended similar results based on what other people also searched for.


keras tensorflow tutorial

All Videos Images News Shopping More Settings Tools

About 960,000 results (0.48 seconds)

Keras as a simplified interface to TensorFlow: tutorial - The Keras Blog
https://blog.keras.io/keras-as-a-simplified-interface-to-tensorflow-tutorial.html
Apr 24, 2016 - Note that this **tutorial** assumes that you have configured **Keras** to use the **TensorFlow** backend (instead of Theano). Here are instructions on ...

People also search for

tf.keras vs keras	convert tensorflow model to keras
keras tensorflow tutorial pdf	convert trained keras to tensorflow
keras tensorflow backend	force keras to use gpu

Related search
best books on neural networks

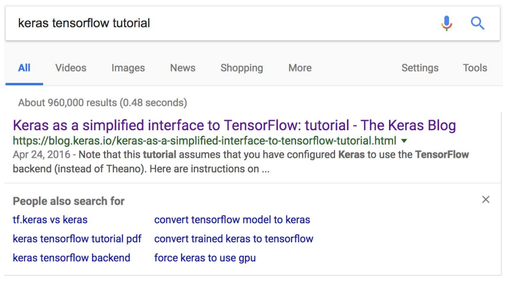
Learning TensorFlow: A Guide t... · TensorFlow for Deep Learning... · Machine Learning with Tens... · Deep Learning with Python · Deep Learning with Keras

Feedback

Searches related to keras tensorflow tutorial

keras tensorflow tutorial **pdf**	keras tutorial **pdf**
keras tutorial	keras **vs** tensorflow
keras tensorflow tutorial **mnist**	keras tutorial **github**
tensorflow keras	**tf.keras** tutorial

Or, if you make it to the bottom of the page, you'll see related searches, in this case suggesting "best books on neural networks"


keras tensorflow tutorial

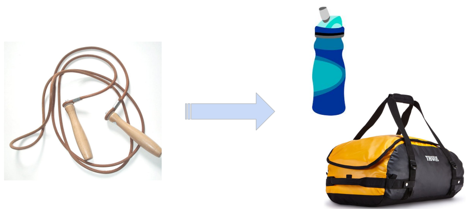
All Videos Images News Shopping More Settings Tools

About 960,000 results (0.48 seconds)

Keras as a simplified interface to TensorFlow: tutorial - The Keras Blog
https://blog.keras.io/keras-as-a-simplified-interface-to-tensorflow-tutorial.html
Apr 24, 2016 - Note that this **tutorial** assumes that you have configured **Keras** to use the **TensorFlow** backend (instead of Theano). Here are instructions on ...

People also search for

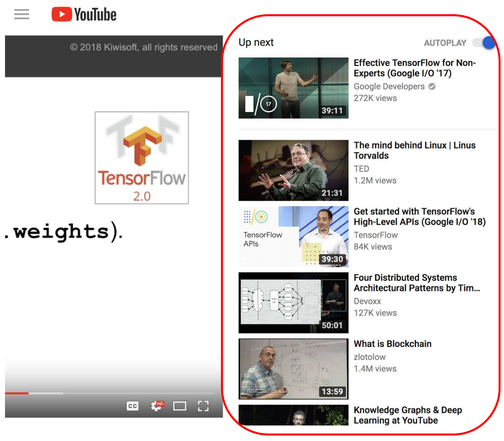
tf.keras vs keras	convert tensorflow model to keras
keras tensorflow tutorial pdf	convert trained keras to tensorflow
keras tensorflow backend	force keras to use gpu



This same kind of recommendation is working behind the scenes when, say, you're buying a jump rope and you're recommended to buy a water bottle or gym bag along with it.

These recommenders have learned that certain items go together, and customers who've bought one of these things are likely to benefit from buying the other.

https://commons.wikimedia.org/wiki/File:LeatherSkippingRope.JPG
https://pixabay.com/en/water-bottle-graphic-bottle-2238811/
https://commons.wikimedia.org/wiki/File:Thule_Chasm_Zinnia_(11726732813).jpg



© 2018 Kiwisoft, all rights reserved

.weights).

Up next AUTOPLAY

Effective TensorFlow for Non-Experts (Google I/O '17)
Google Developers
272K views
39:11

The mind behind Linux | Linus Torvalds
TED
1.2M views
21:31

Get started with TensorFlow's High-Level APIs (Google I/O '18)
TensorFlow
84K views
39:30

Four Distributed Systems Architectural Patterns by Tim...
Devoxx
127K views
50:01

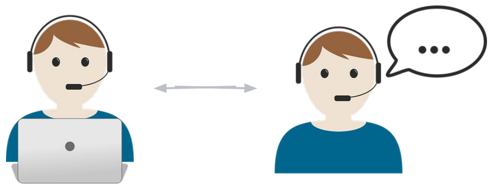
What is Blockchain
zlotolow
1.4M views
13:59

Knowledge Graphs & Deep Learning at YouTube

Recommender systems also provide a way to keep us engaged.

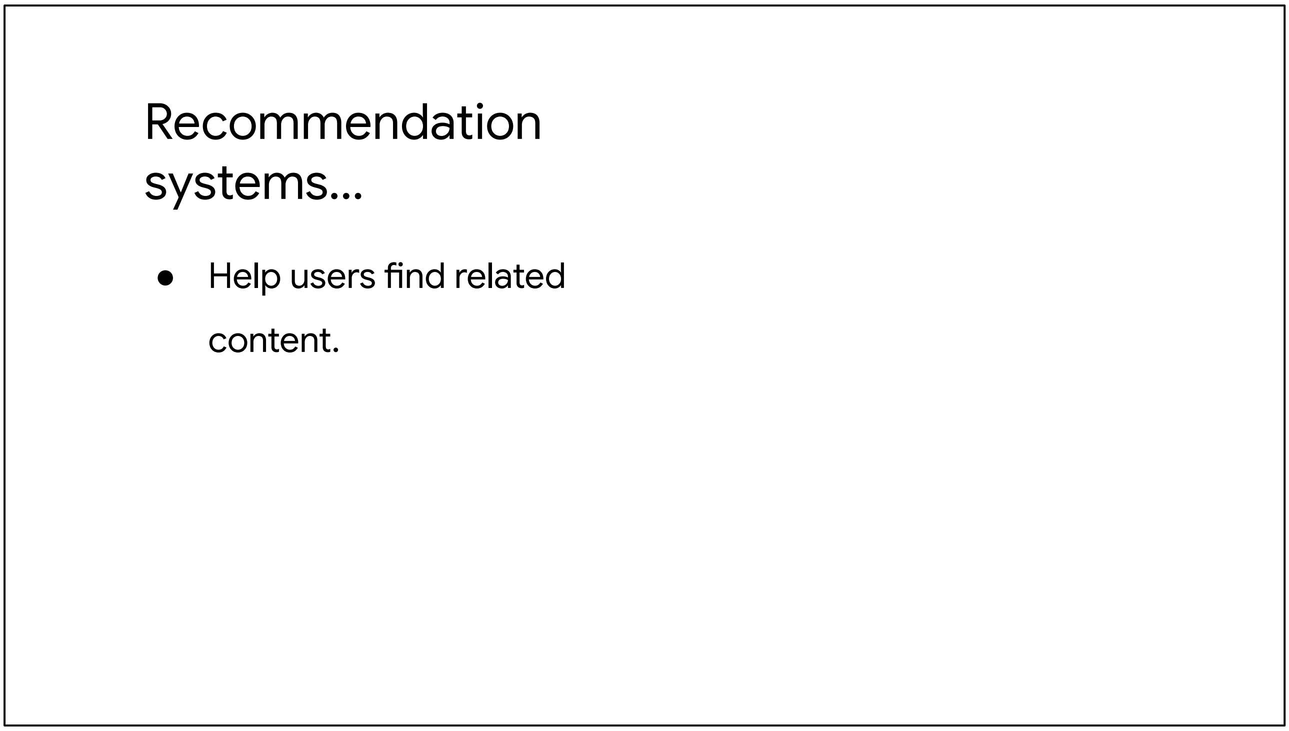
When you watch a video on YouTube, you've probably noticed the banner just to the right of your video that contains a list of recommended videos that are "Up Next." When your video finishes, these new ones will play—and we'd like them to be interesting and useful for you.

This serves two purposes: it keeps the user happy and interested and also keeps them engaged with the product.



Even outside the realm of products, recommender systems have become an important way for businesses to interact with their customers. For example, when you contact a call center, a recommendation engine can help ensure that your query is directed to the appropriate division, which improves their operations and keeps customers happy.

https://pixabay.com/en/support-help-hotline-headset-1984615/



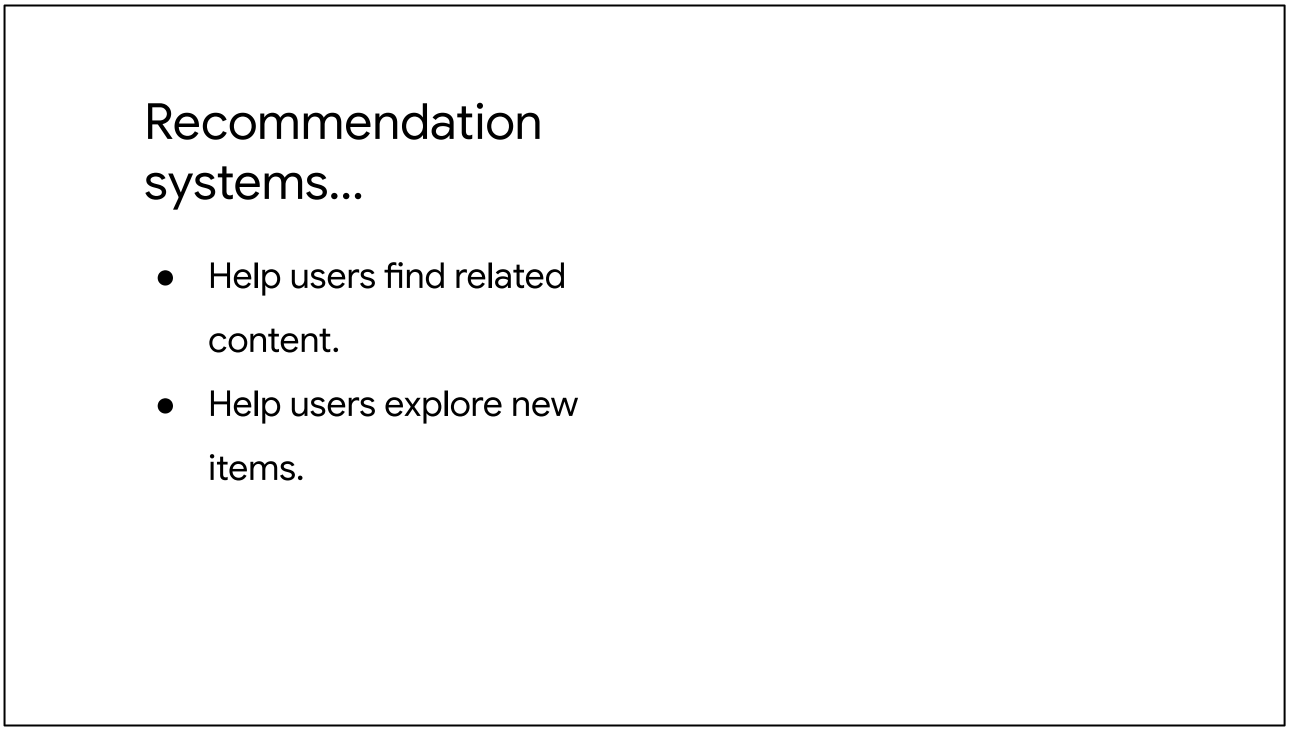
Recommendation systems...

- Help users find related content.

In all of these examples, recommender systems provide benefit for both people who provide them and the people who use them.

For users, they help us explore and understand the item space.

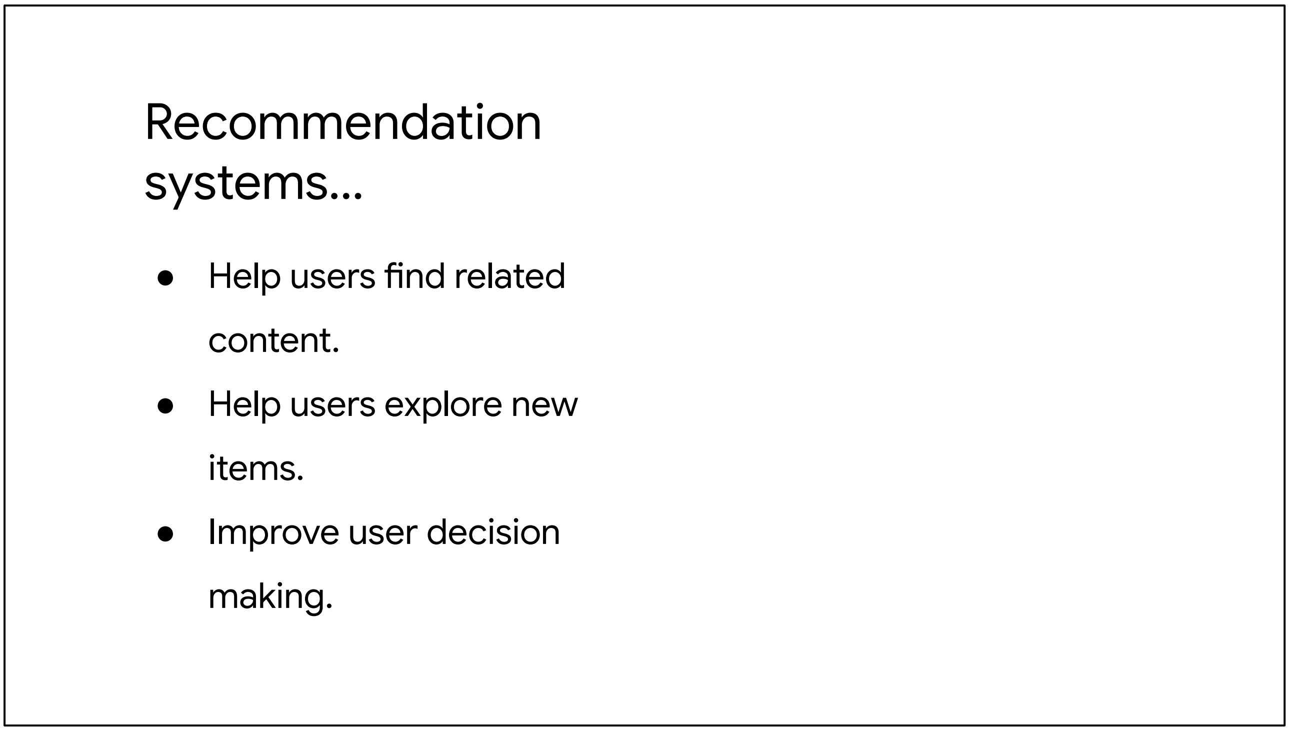
Whether that is by helping us find related content......



Recommendation systems...

- Help users find related content.
- Help users explore new items.

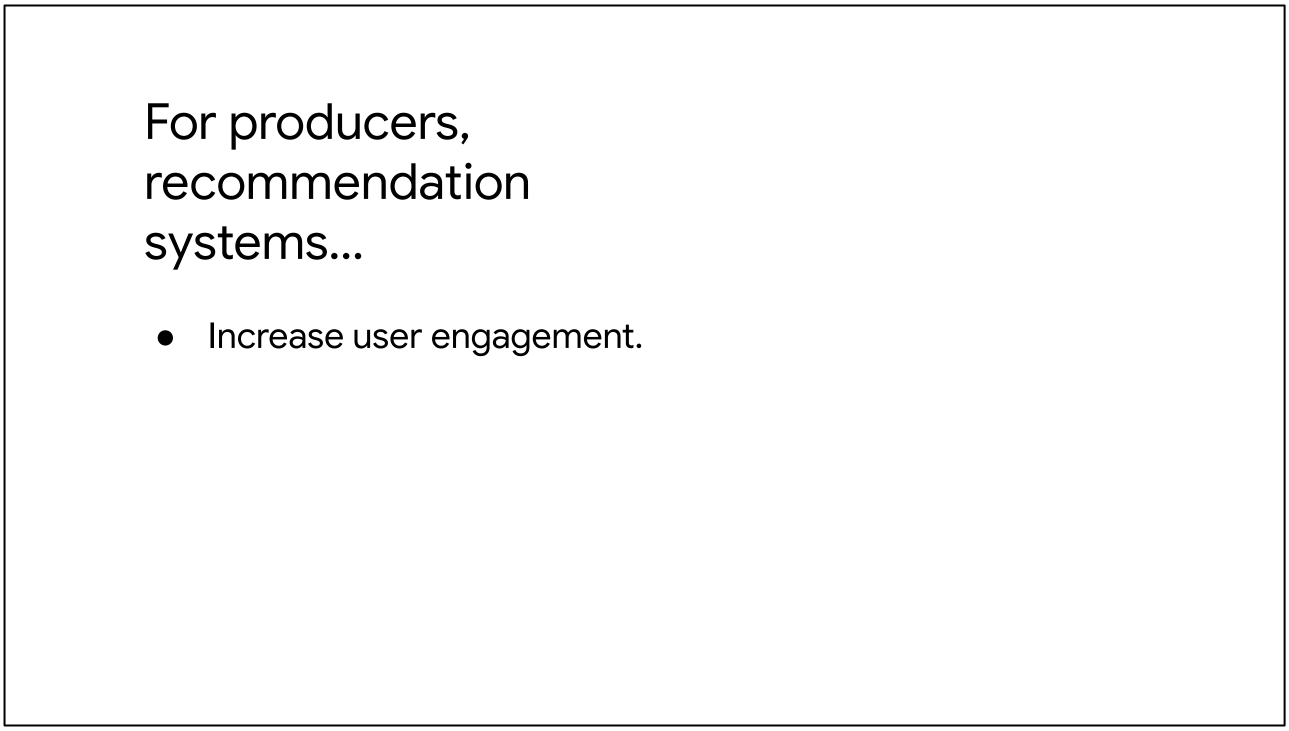
.....or helping us explore new items entirely.



Recommendation systems...

- Help users find related content.
- Help users explore new items.
- Improve user decision making.

They also improve decision making by decreasing a potentially massive item space to only those few items relevant to the user, and much more.



For producers, recommendation systems...

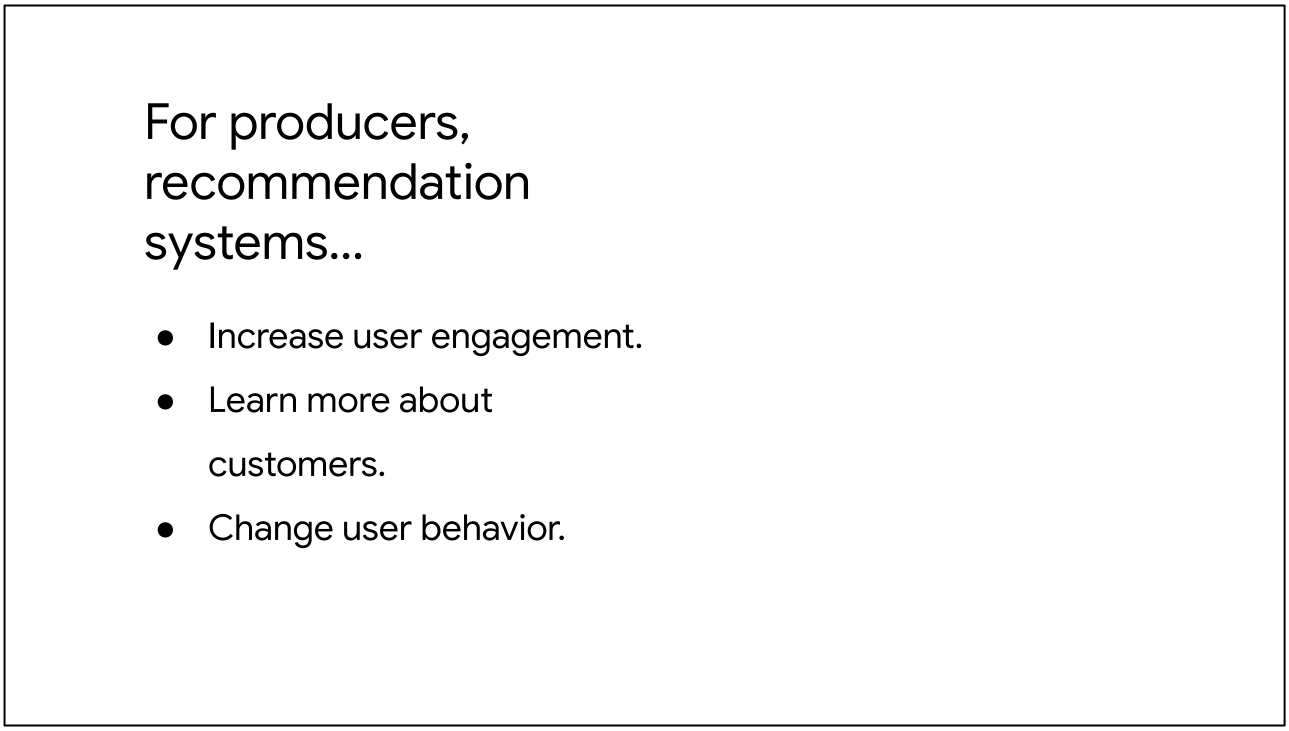
- Increase user engagement.

From the recommender provider perspective,
they increase user satisfaction and engagement, which leads to deeper customer loyalty and trust.

For producers, recommendation systems...

- Increase user engagement.
- Learn more about customers.

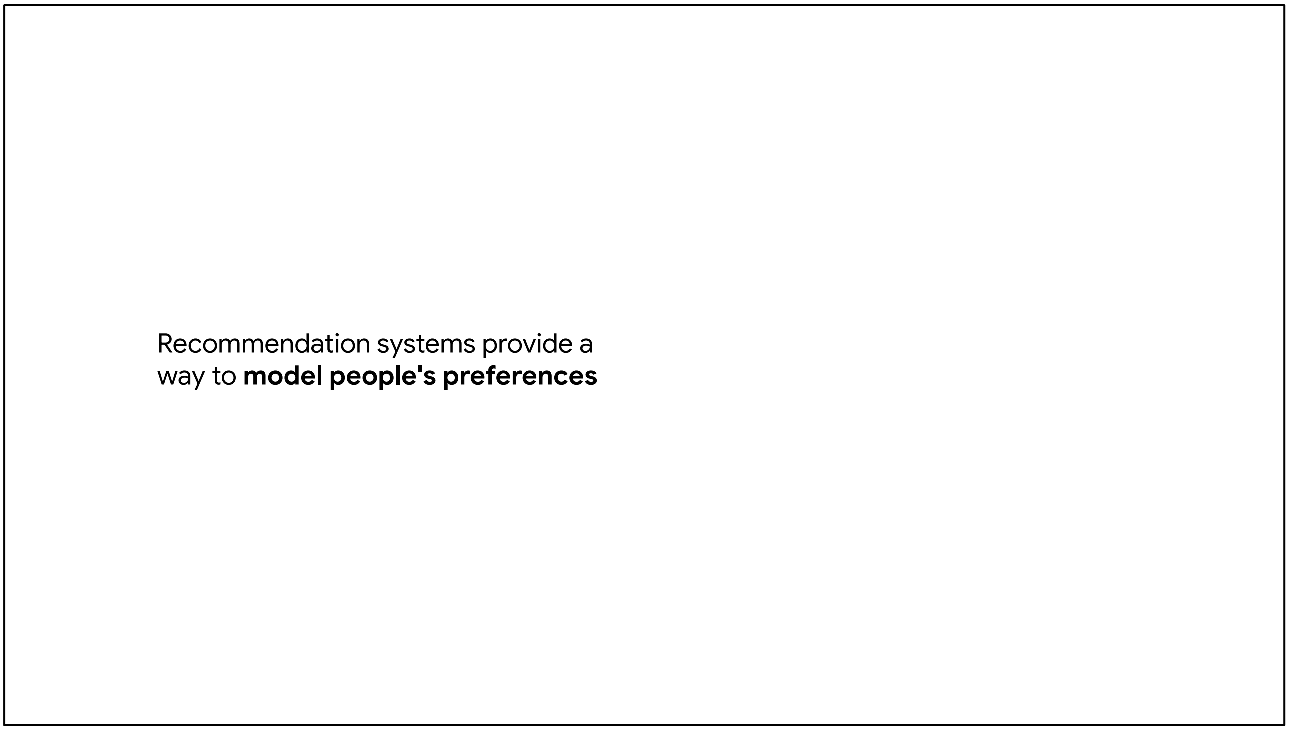
They also allow providers to learn more about their customers by examining their interactions with the recommended suggestions.



For producers, recommendation systems...

- Increase user engagement.
- Learn more about customers.
- Change user behavior.

Or even change user behavior and drive demand towards less popular, but still relevant, items for a user.



Recommendation systems provide a way to **model people's preferences**

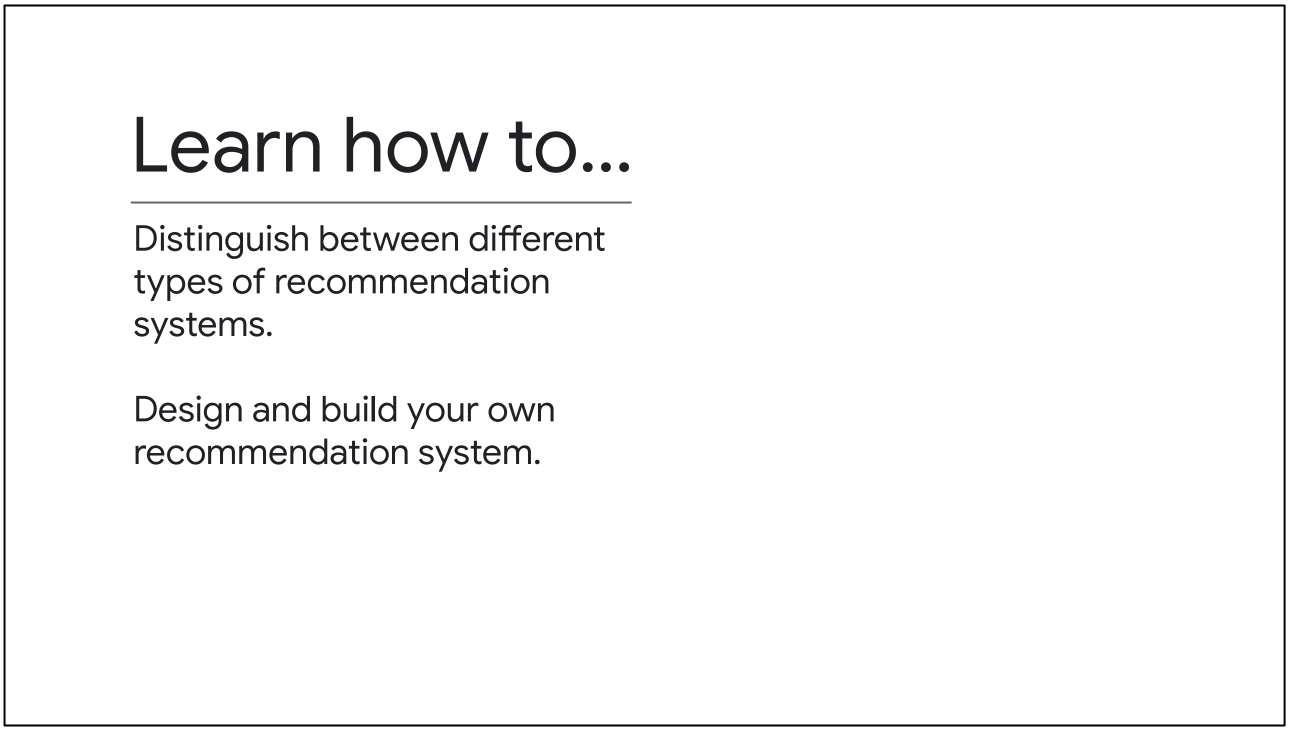
In short, recommender systems provide a way to model people's preferences and behaviors, which allows providers to reach the right customer efficiently and effectively, with the right message at the right time.

Learn how to...

Distinguish between different types of recommendation systems.

In the upcoming modules we will cover some of the details of recommendation engines.
We'll discuss different types of popularly used recommendation engines, including content-based and collaborative filtering, knowledge-based recommenders, and deep learning techniques.

We'll do a deep dive into some of these approaches in later modules.



Learn how to...

Distinguish between different types of recommendation systems.

Design and build your own recommendation system.

We will also see how you can design and build your own recommendation engine, following each of these methods.

Learn how to...

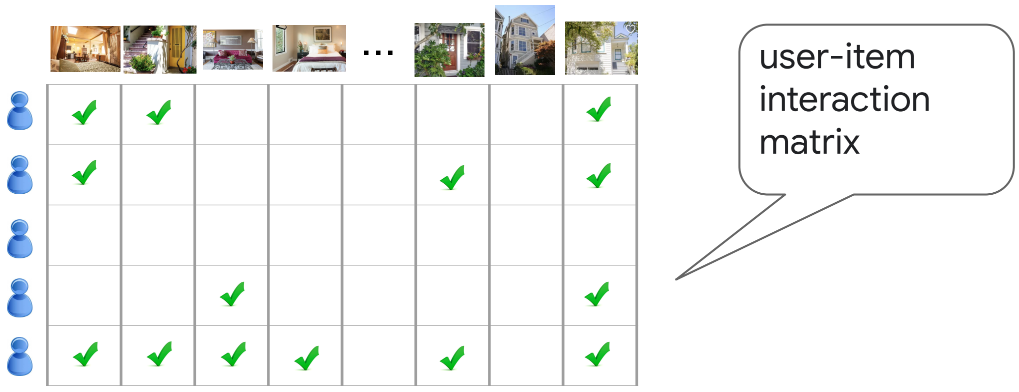
Distinguish between different types of recommendation systems.

Design and build your own recommendation system.

Anticipate common problems that arise when developing recommendation systems.

We'll end this module by discussing some of the common pitfalls that arise when developing recommender systems and how they can be addressed.

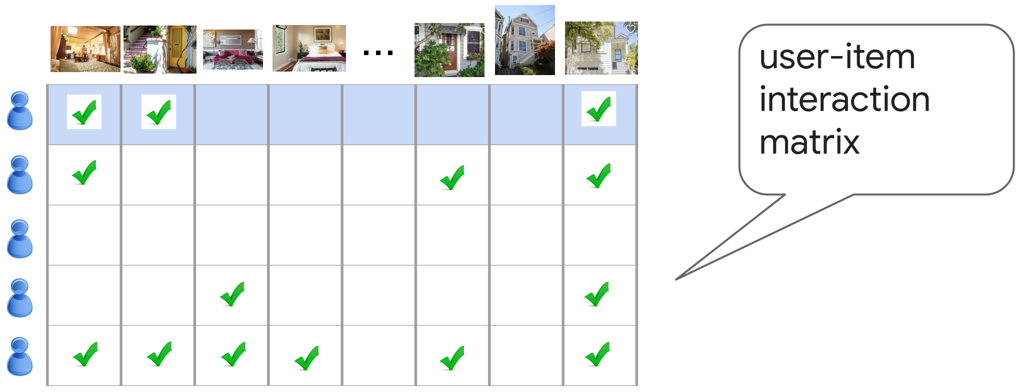
How would you recommend a vacation house to a user?



To get started, let's consider a thought exercise.
Suppose you wanted to build an application to suggest houses for people who want to rent a vacation home.
You have a database of users and properties and information about users' past rentals, various property details, and the user's respective ratings.

We can represent this information in a matrix like the one here, called a user-item interaction matrix or user-item matrix

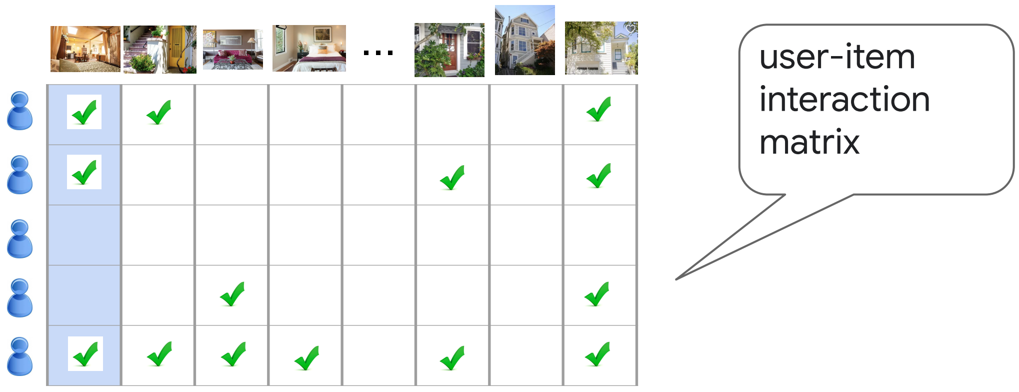
How would you recommend a vacation house to a user?



✓	✓						✓
✓					✓		✓
		✓					✓
✓	✓	✓	✓		✓		✓

Each row corresponds to a user.
Users could be customers, visitors, app users, readers. Here we have just 5, but we could have millions or even billions of users
This user has rated 3 of the houses in our database.

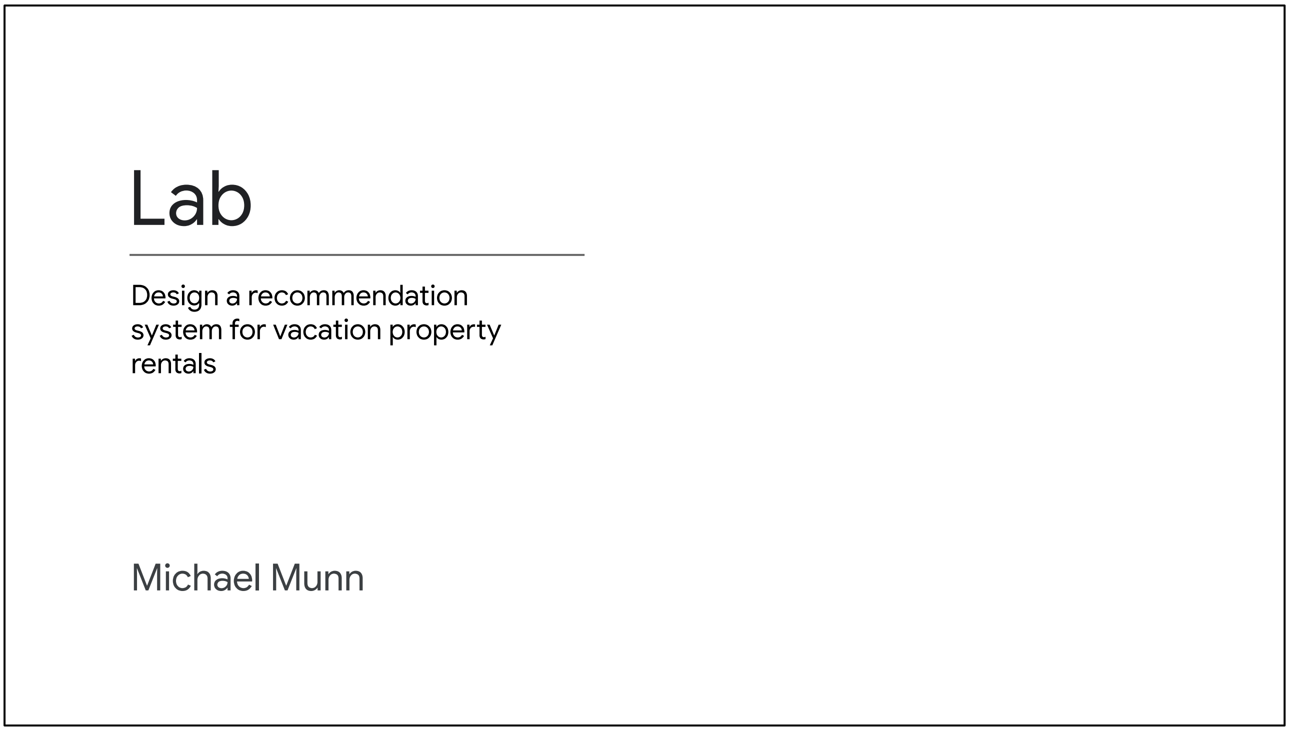
How would you recommend a vacation house to a user?



				…			
✓	✓						✓
✓					✓		✓
		✓					✓
✓	✓	✓	✓		✓		✓

Each column corresponds to a item.
Items could be products, movies, events, articles; we could have thousands.
In this case, items are properties for rent. This item has been rated by three separate users.

In general, if user i has a rating for house j, then we have a score/checkmark in the (i,j) spot.

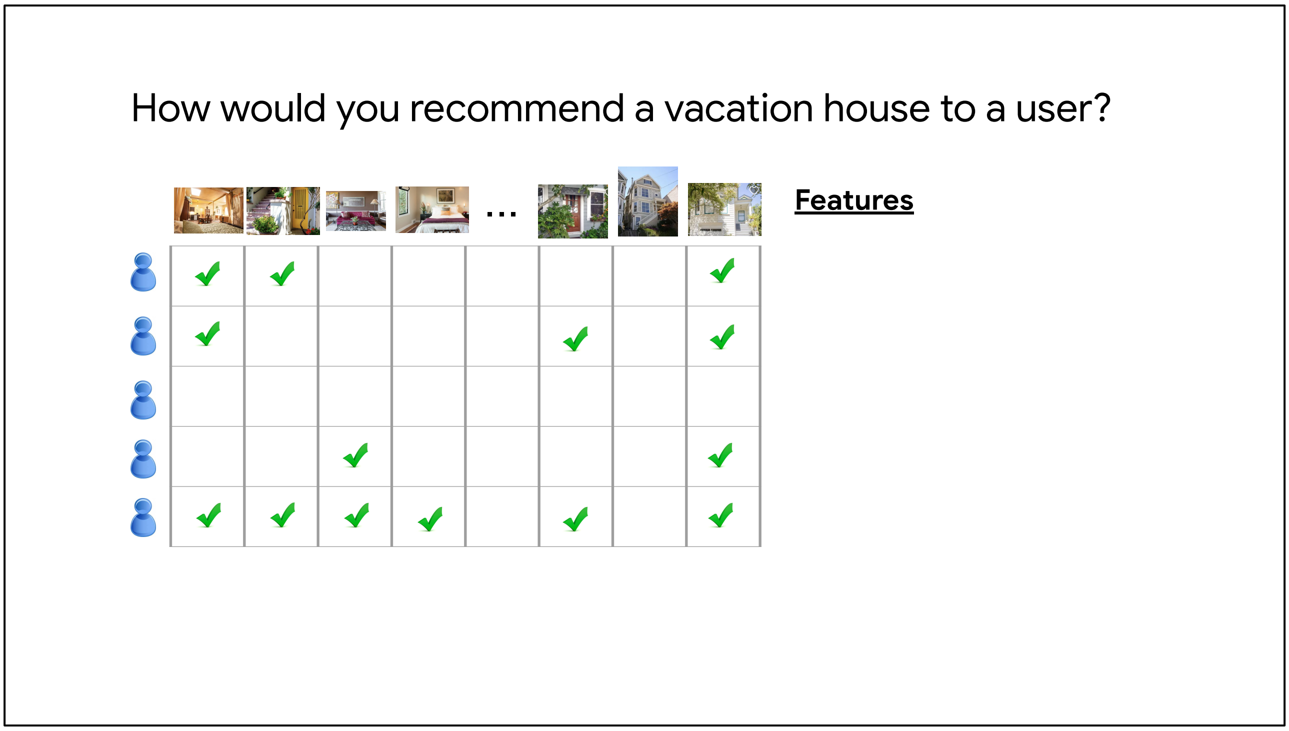


Lab

Design a recommendation system for vacation property rentals

Michael Munn

Take a few minutes and think about how you would go about selecting a vacation rental to recommend to a user?
What features will you use?
What target or label will you try to predict? Where will your data come from?
Try to be as thorough as possible.

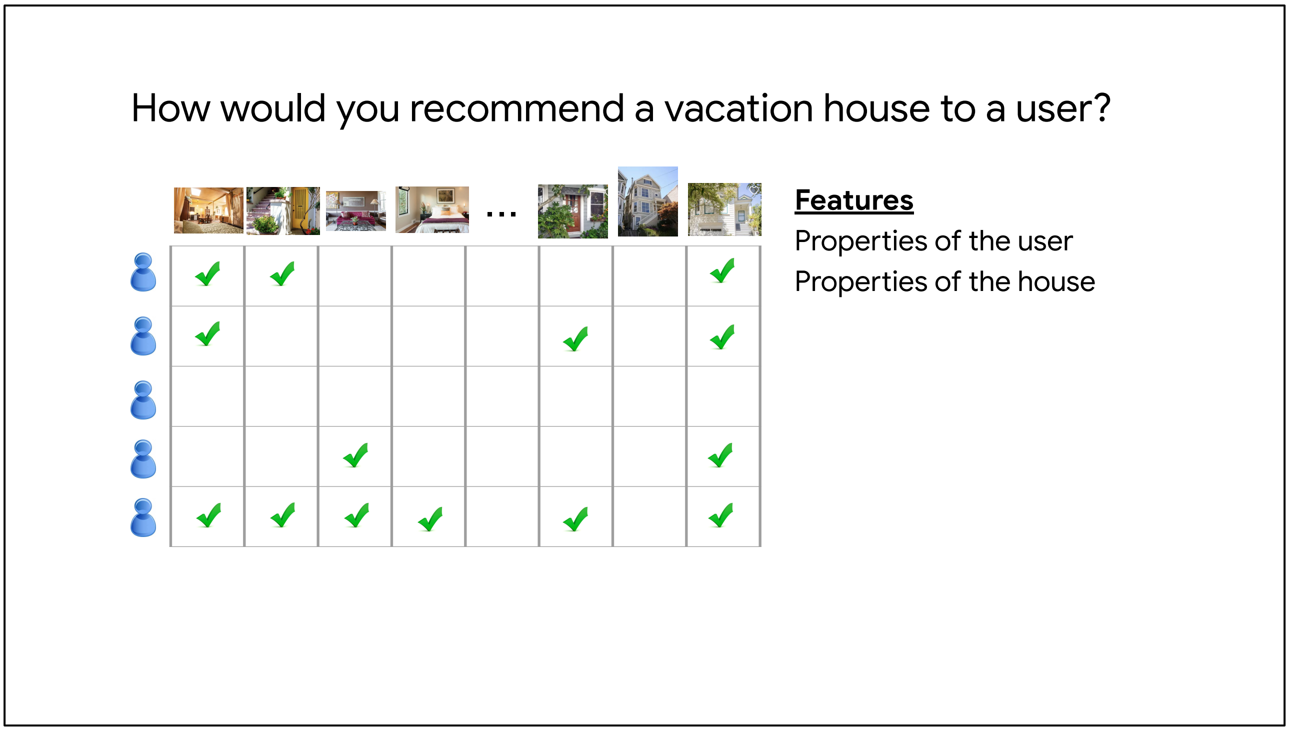


How would you recommend a vacation house to a user?

					…			
	✓	✓						✓
	✓					✓		✓
			✓					✓
	✓	✓	✓	✓		✓		✓

Features

What did you come up with?
What features did you think would be relevant?
...



How would you recommend a vacation house to a user?

				...			
✓	✓						✓
✓					✓		✓
		✓					✓
✓	✓	✓	✓		✓		✓

Features

Properties of the user

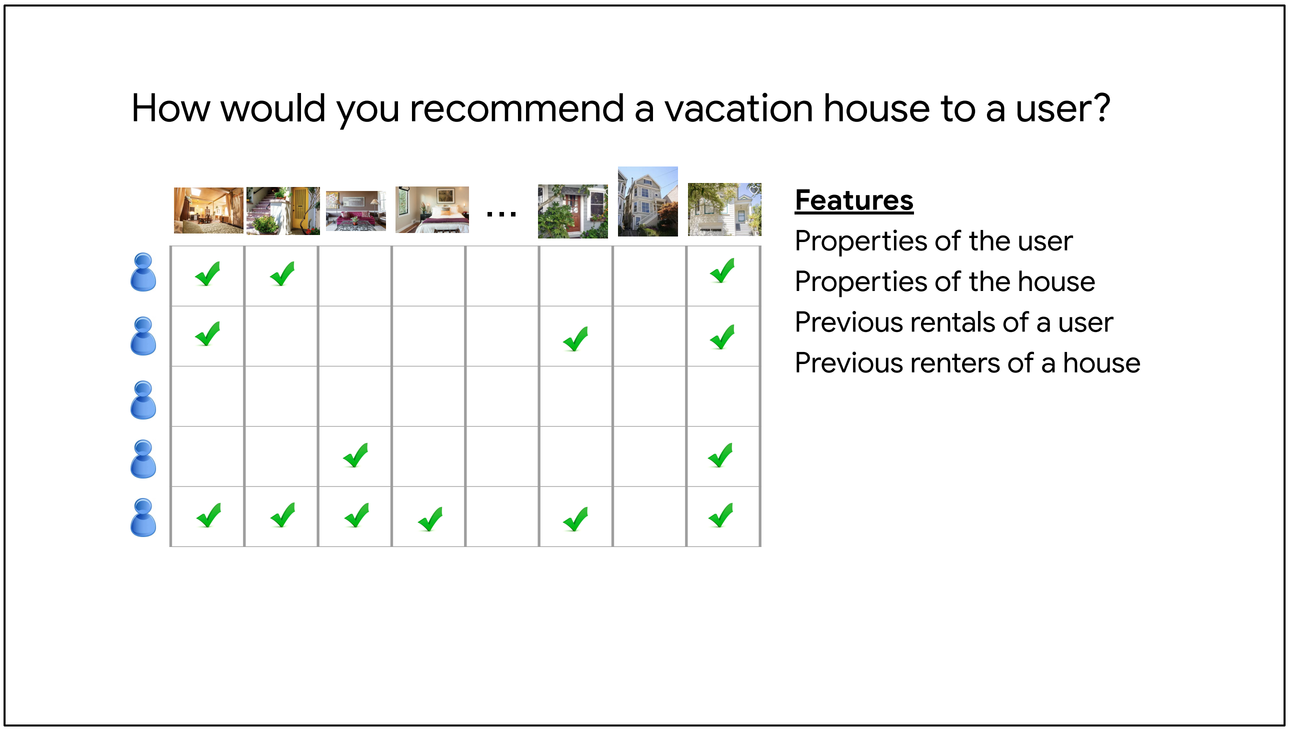
Properties of the house

...

Did you think about ways to use properties of the user? If so, what features did you use to describe your users?

Did you consider using properties of the house? What features did you use to categorize the houses?

...

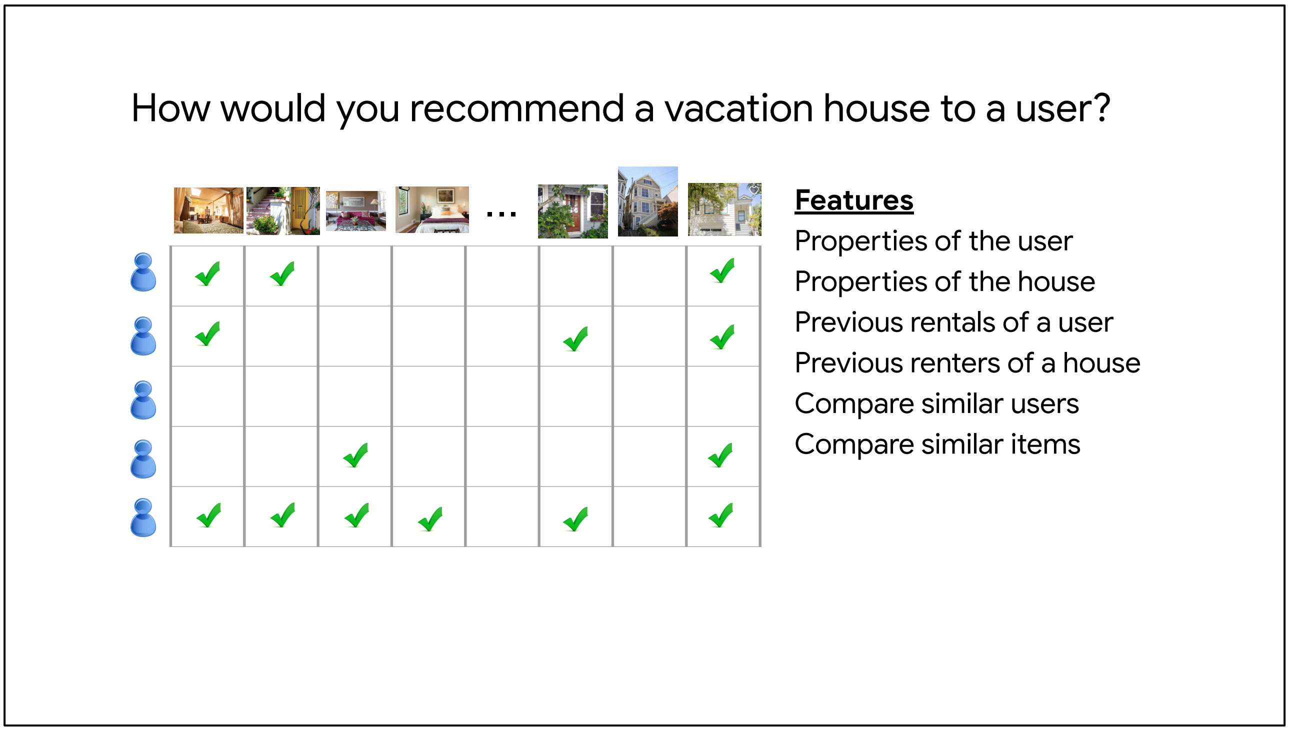


How would you recommend a vacation house to a user?

					…			
	✓	✓						✓
	✓					✓		✓
			✓					✓
	✓	✓	✓	✓		✓		✓

Features
Properties of the user
Properties of the house
Previous rentals of a user
Previous renters of a house

...
Maybe you thought to consider a user's previous rentals.
Or to consider the previous renters of a given house.
...



How would you recommend a vacation house to a user?

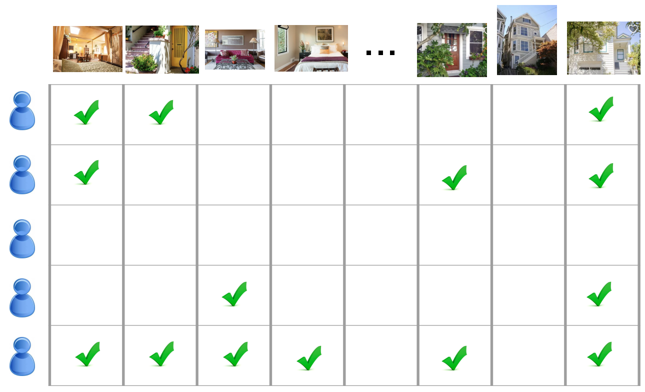
				…			
✓	✓						✓
✓					✓		✓
		✓					✓
✓	✓	✓	✓		✓		✓

Features
Properties of the user
Properties of the house
Previous rentals of a user
Previous renters of a house
Compare similar users
Compare similar items

...
Would it be useful to compare similar users' behavior to find new recommendations?
Or to suggest similar properties that a user might like?

What does it mean for two users to be similar? What does it mean for two properties to be similar?
How do you measure similarity for things like this?

How would you recommend a vacation house to a user?



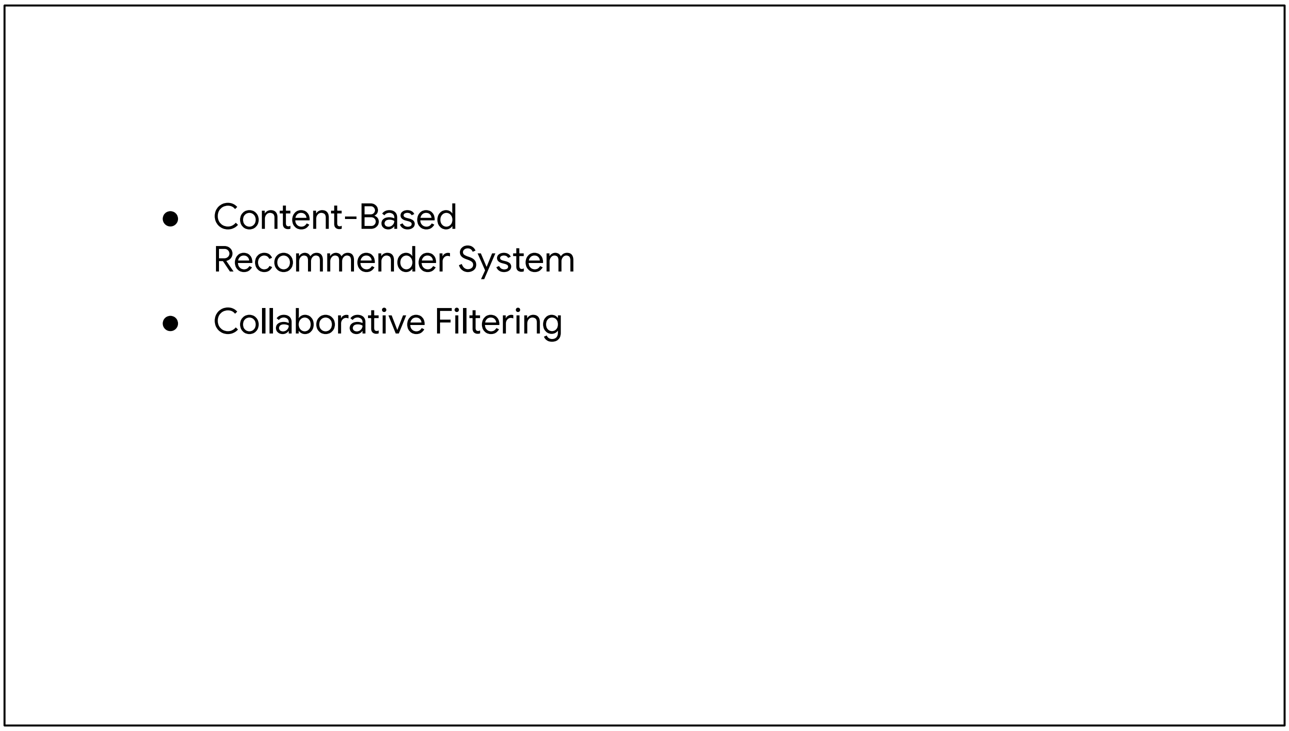
Features

Properties of the user
Properties of the house
Previous rentals of a user
Previous renters of a house
Compare similar users
Compare similar items

Targets

Rating for unseen properties
Next rental property

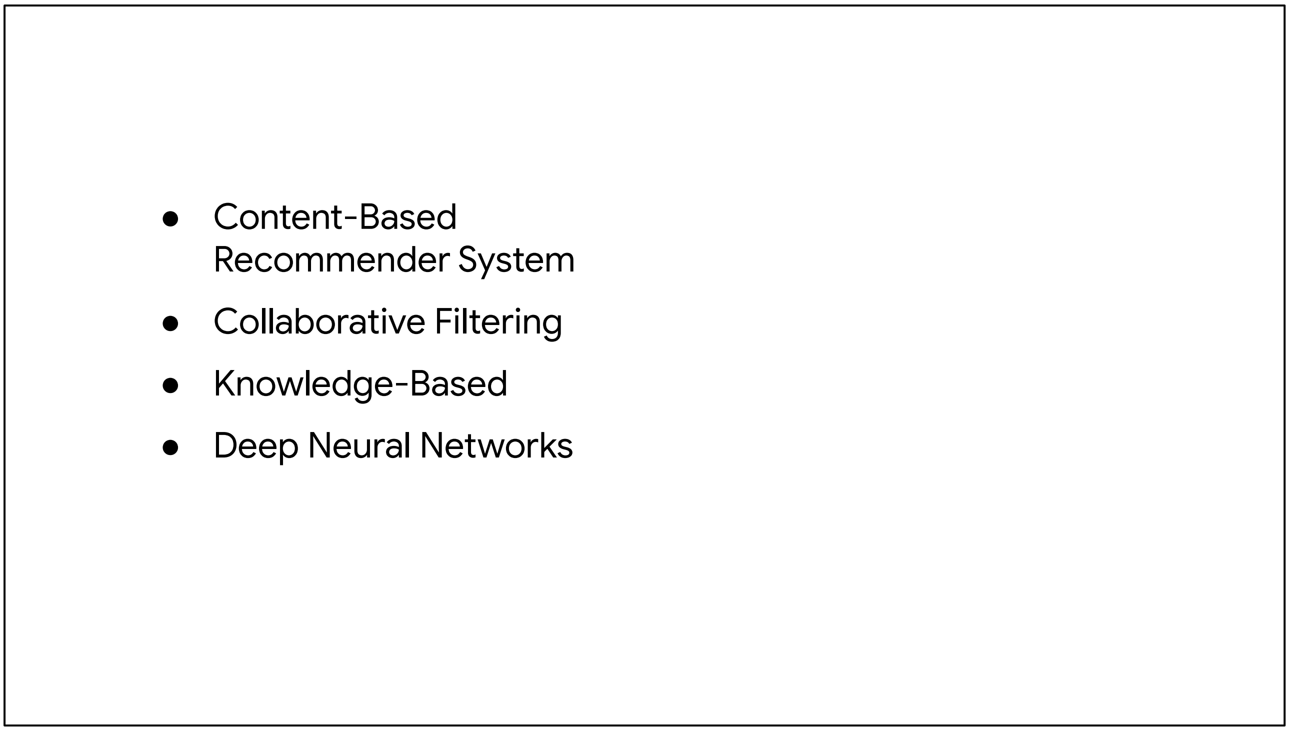
What were you trying to model; that is, what label did you think to use?
Did you try to predict a user's rating score for a new property?
Or perhaps you wanted to simply predict what house they would book next.



- Content-Based Recommender System
- Collaborative Filtering

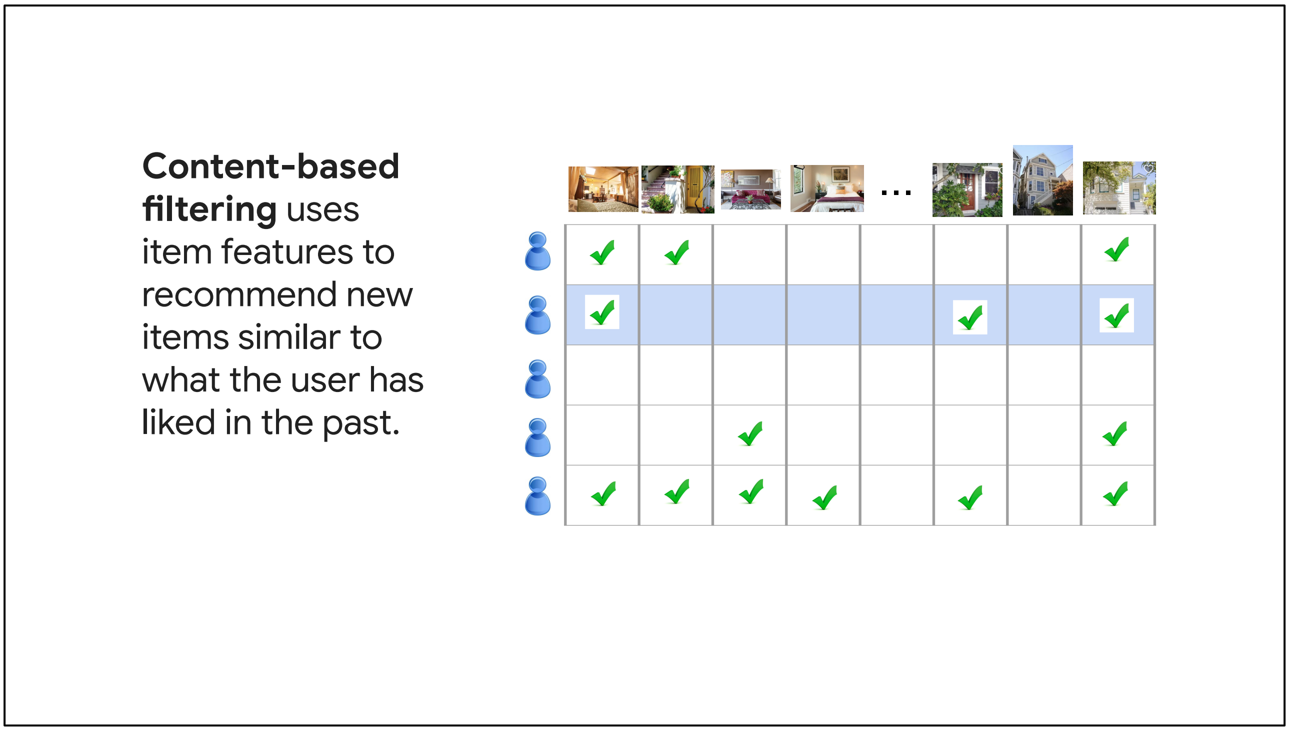
Different techniques of recommendation engines approach these questions in different ways, and we will briefly introduce a few of these approaches here.

We will start with two of the most popular types of recommendation engines: content-based and collaborative filtering.



- Content-Based
  Recommender System
- Collaborative Filtering
- Knowledge-Based
- Deep Neural Networks

We'll also briefly discuss knowledge-based recommenders and introduce how deep learning can be used in building a recommendation engine, discussing along the way some of the common pain points that arise.



Content-based filtering uses item features to recommend new items similar to what the user has liked in the past.

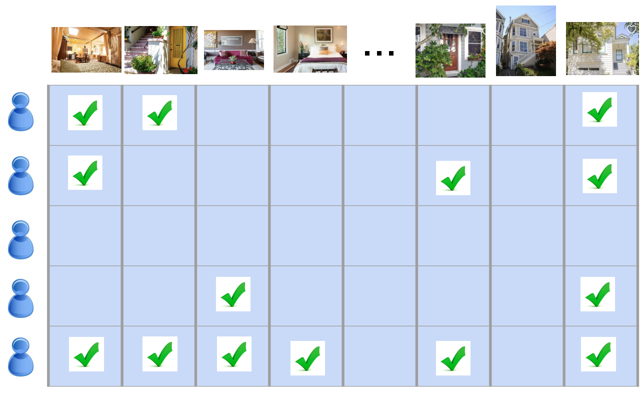
					...			
	✓	✓						✓
	✓					✓		✓
			✓					✓
	✓	✓	✓	✓		✓		✓

A content-based method uses attributes of the items to recommend new items to a user.
It doesn't take into account the behavior or ratings of other users.

For example, if a user has rented and liked a lot of vacation homes on the beach, this method will suggest other similar homes that are also on the beach.
This is often done by hand-engineering features for the items and learning how much a single user aligns with each of those features.

Using that feature representation of the user, it is then possible to extrapolate how a given user would rank unseen items.

Collaborative Filtering uses similarities between users and items simultaneously to determine recommendations.

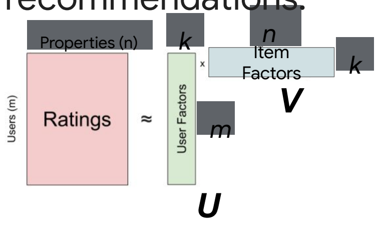


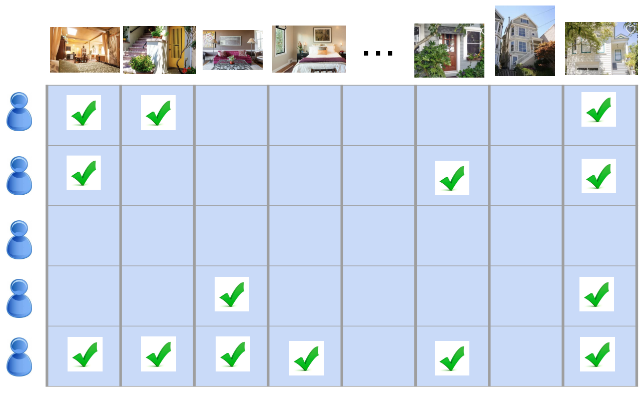
A collaborative filtering model works with the entire user-item interaction matrix. They consider all users, all items, and all user-item ratings.

Loosely speaking, they work with the idea that similar users will like similar items. That is, they use similarities between users and items simultaneously to provide recommendations.
This can allow for seemingly serendipitous recommendations; meaning, they can recommend an item to user A based on the interests of a similar user B.

Another useful advantage is that the feature representations can be learned automatically, so you don't have to rely on hand-engineering specific features as you might for content-based filtering.

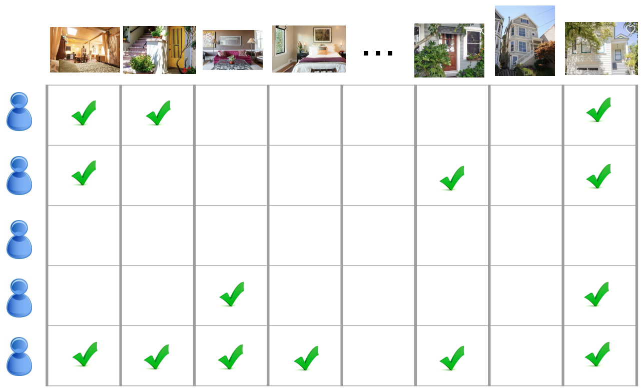
Collaborative Filtering uses similarities between users and items simultaneously to determine recommendations.





This process often involves matrix factorization and behaves similarly to a content-based approach, but does not rely on previously constructed features.

Knowledge-based recommender systems use explicit knowledge about the users, items, and recommendation criteria.



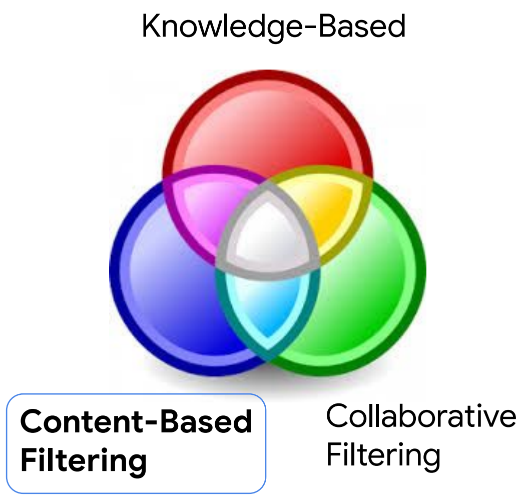
					…			
	✓	✓						✓
	✓					✓		✓
			✓					✓
	✓	✓	✓	✓		✓		✓

Knowledge-based recommender systems are based on explicit knowledge about the user preferences, items, and/or recommendation criteria.
They are especially useful when alternative approaches, such as collaborative filtering or content-based methods, cannot be applied. This occurs in situations where items are not purchased very often.

https://pixabay.com/en/realtor-real-estate-3261160/

For example, if instead of renting a vacation house, suppose we wanted to build a recommendation engine for *buying* a vacation house. Because most people don't buy houses often, we probably wouldn't have enough previous house-buying information to use either a content-based filtering or collaborative filtering approach.

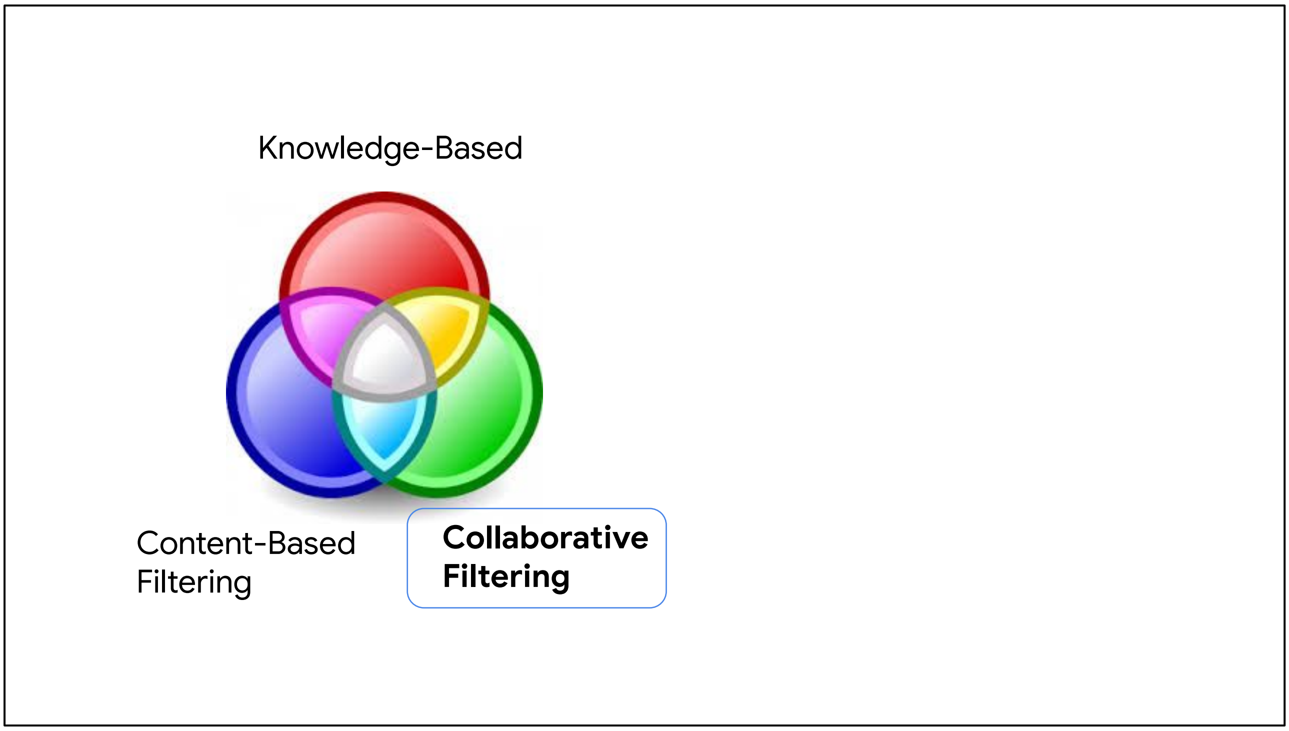
In this scenario, knowledge-based systems will often explicitly ask users for their preferences and then use that information to begin making recommendations.



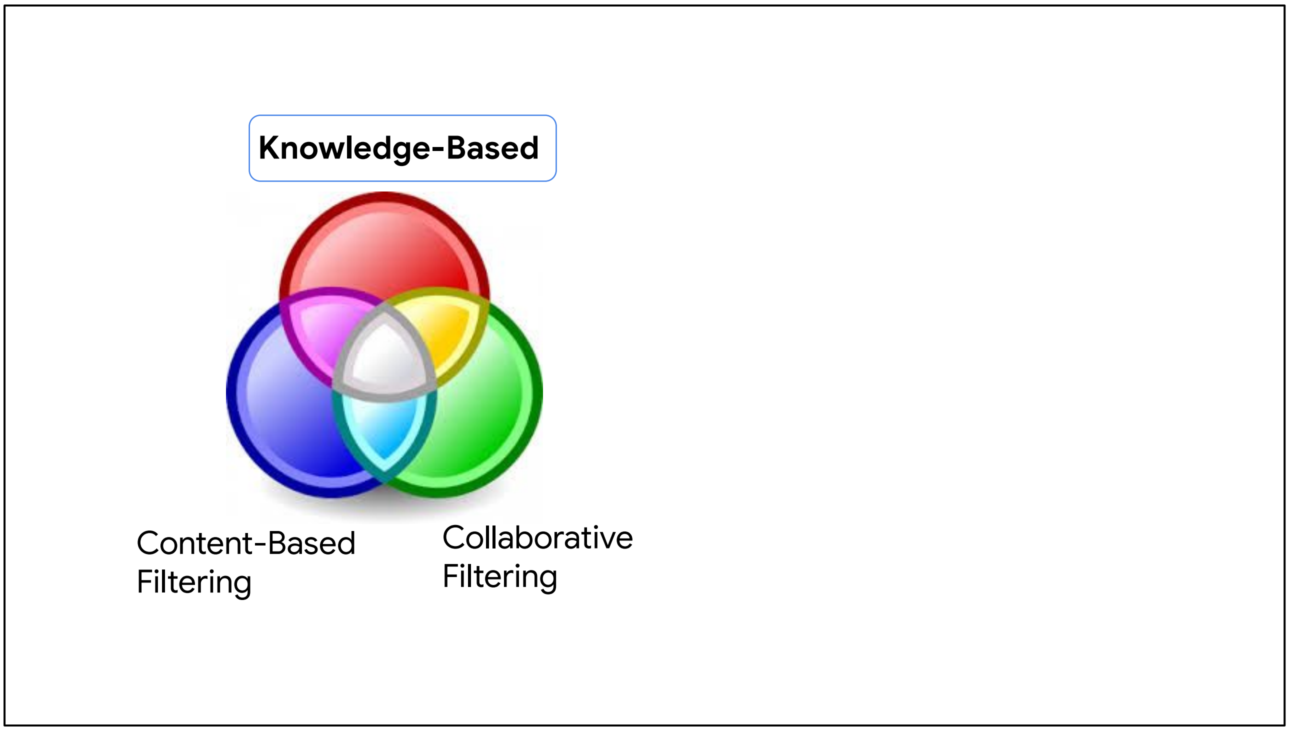
Often there is value in combining different types of recommendation models into a single hybrid approach.
This can be done in several ways.

For example, we could develop a few recommenders and then use one or the other, depending on the scenario.
If a user has already rated a large number of items, perhaps we can rely on a content-based method.

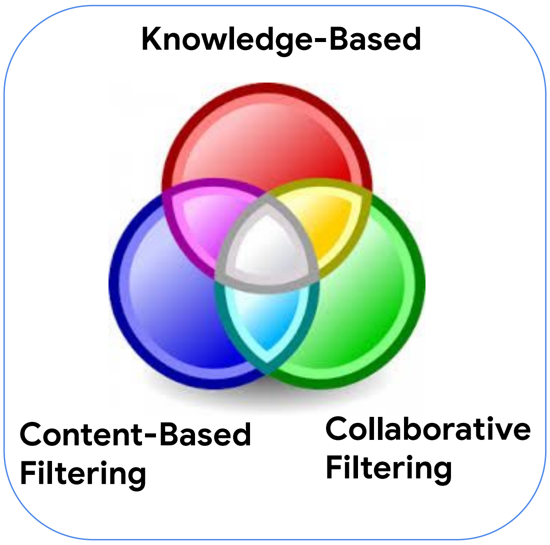
https://pixabay.com/en/venn-diagram-set-diagram-diagrams-41219/



However, if the user has rated only a few items, we may instead prefer to use our collaborative filtering approach.
This way we can fully leverage the information we have about other users and their interactions with items in our database to gain some insight into what we can recommend.



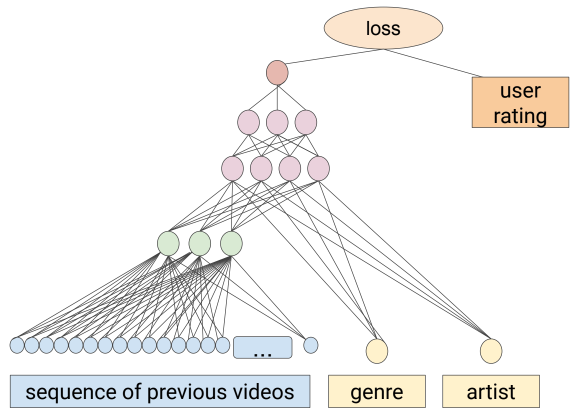
Of course, if we have no information about a user's previous item interactions or we lack any information about a given user, we may instead want to rely on a knowledge-based approach and ask the user directly for their preferences via a survey before making recommendations.



Another way to create a hybrid model is to simply combine the outcomes of more than one of these methods. The multiple outcomes could then form the input to a more sophisticated model that makes the final recommendation that we serve to the user.

The idea is that the more sophisticated model will learn a more nuanced relationship between the query and the various model outcomes, and we'll have much better recommendations. In fact, some research suggests that a hybrid approach—combining multiple outcomes like this—can provide more accurate recommendations than a single approach on its own.

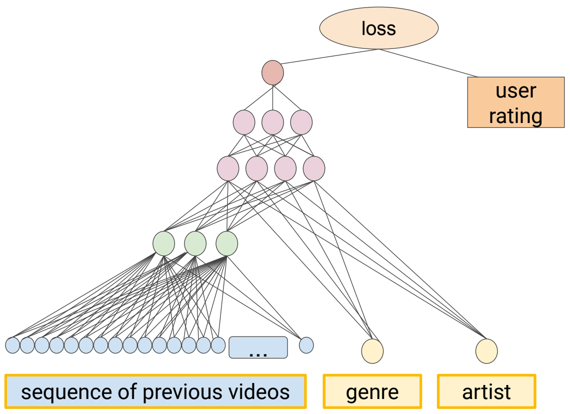
Deep neural networks can be trained to predict ratings based on user and item attributes.



In addition to content-based, collaborative filtering and knowledge-based approaches, deep learning models can also be used when building a recommendation system.

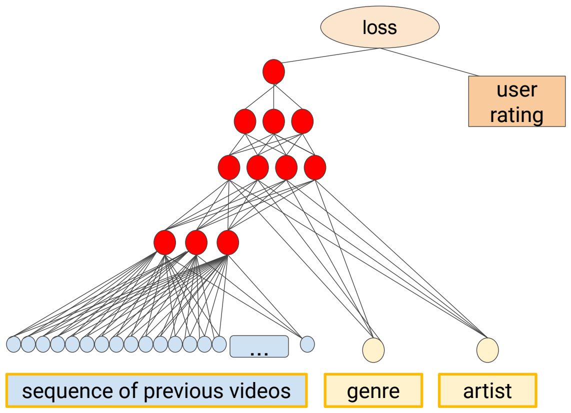
Deep neural networks work well because they are flexible and can be trained to have varying outcomes, such as predicting ratings, interactions, or even next items.

Deep neural networks can be trained to predict ratings based on user and item attributes.

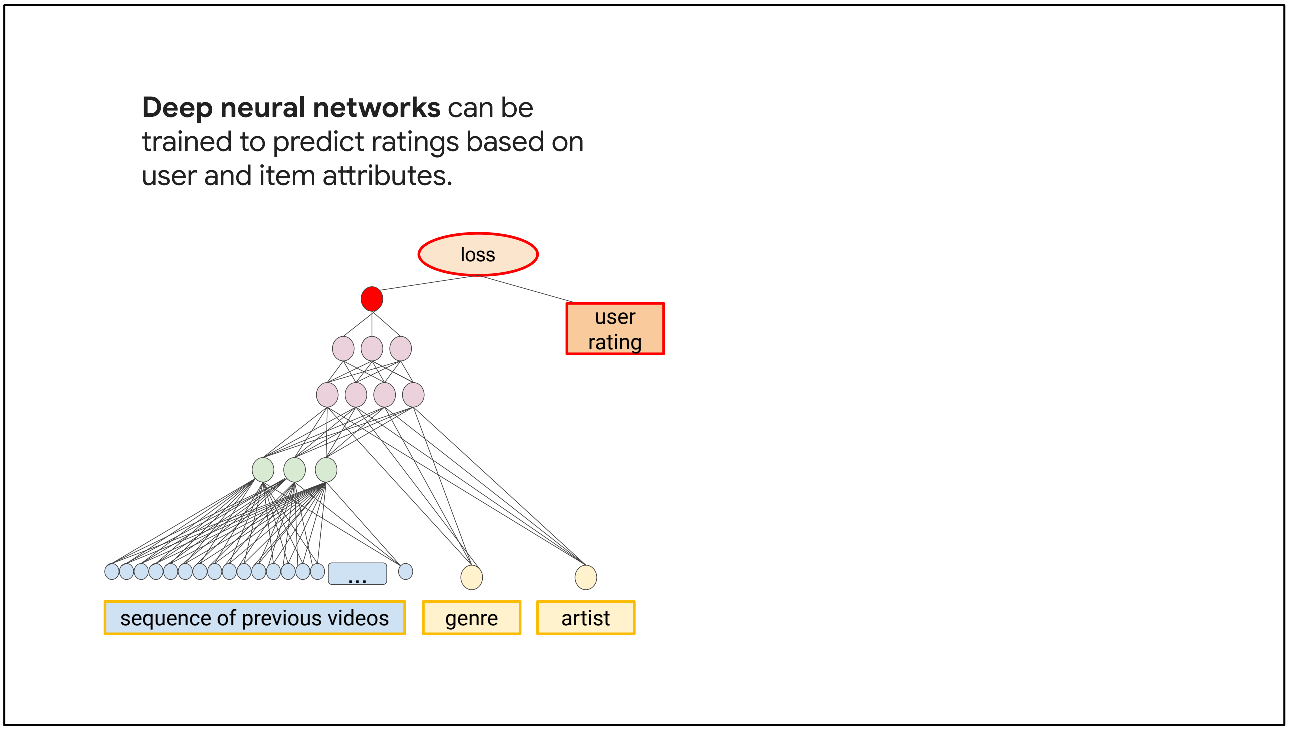


For example, suppose we wanted to recommend videos to our users.
We could approach this from a deep learning point of view by taking attributes of the user's behavior as input; for example, a sequence of their previously watched videos embedded into some latent space, combined with video attributes, like the genre or artist information for a given video.

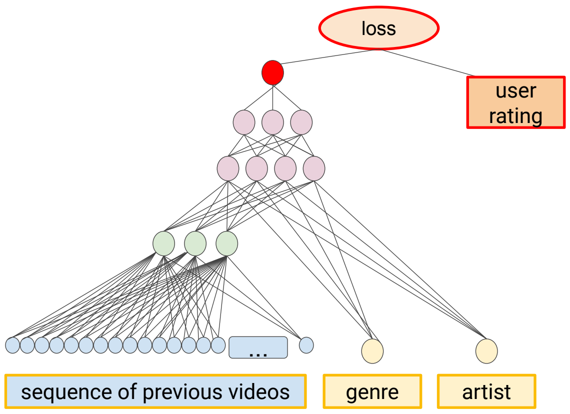
Deep neural networks can be trained to predict ratings based on user and item attributes.



These user and item attributes are combined into a single dense layer of a neural network, and then again through another fully connected layer, until a single value is ultimately produced...

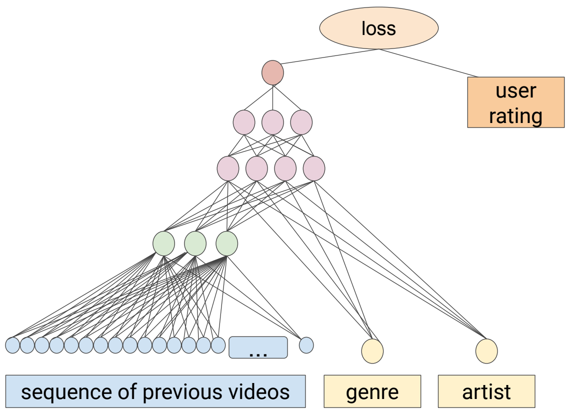


Deep neural networks can be trained to predict ratings based on user and item attributes.



...with the objective function comparing the difference with the user's rating for the given video.
At inference time, we can apply this model to rate previously unseen videos and recommend to the user the video with the highest score.

Deep neural networks can be trained to predict ratings based on user and item attributes.



Deep learning models like this are flexible enough to easily incorporate all kinds of query features and item features into the input layer of the network to help capture the specific interests of a user or to improve the relevance of the recommendations.

Quiz

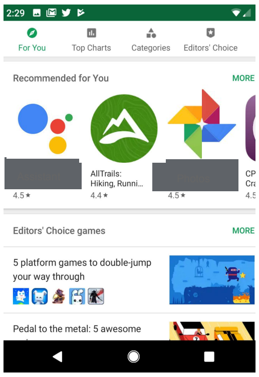
The model recommends a hiking app to a user because they recently installed a similar app. This is an example of what kind of filtering?

a) Content-based filtering

b) Collaborative filtering

c) Deep neural network

d) Hybrid approach



Let's do a quick quiz.
Suppose we have built a recommendation engine to suggest new apps from our app store. Our model recommends a hiking app to a user because they recently installed a running app and have been using it a lot. What kind of recommendation approach is this an example of?

Is it content-based filtering, collaborative filtering, a deep neural network approach, or a hybrid of more than one of these?

Answer

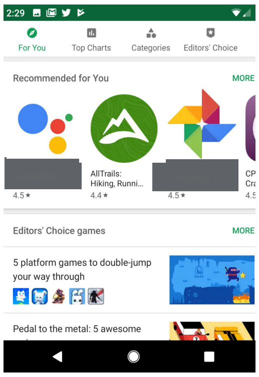
The model recommends a hiking app to a user because they recently installed a similar app. This is an example of what kind of filtering?

[✓ a) Content-based filtering]

b) Collaborative filtering

c) Deep neural network

d) Hybrid approach



That's right, this would be an example of content-based filtering.
We are told that the hiking app suggestion is a result of the user recently installing and using a similar app for tracking their runs. Because the *content* of these two apps is similar, we expect that this user will appreciate both. Hence the name, "content-based filtering."

Answer

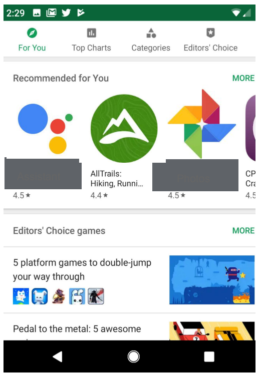
The model recommends a hiking app to a user because they recently installed a similar app. This is an example of what kind of filtering?

[✓ a) Content-based filtering]

b) ~~Collaborative filtering~~

c) Deep neural network

d) Hybrid approach



This wouldn't be collaborative filtering because the recommendation does not rely on the behaviors and item interactions of other users. Its recommendation is based only on that user's previous behavior. This is what sets content-based filtering apart from collaborative filtering.

Answer

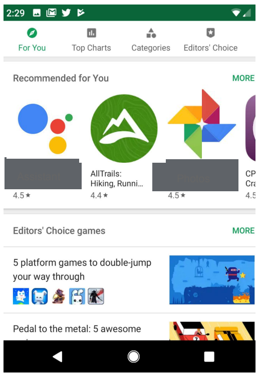
The model recommends a hiking app to a user because they recently installed a similar app. This is an example of what kind of filtering?

[✓ a) Content-based filtering]

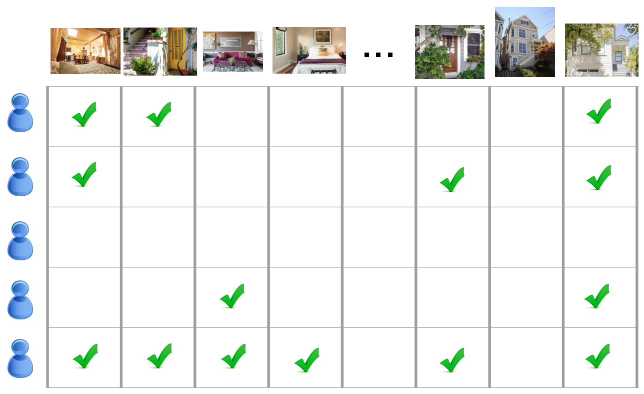
~~b) Collaborative filtering~~

~~c) Deep neural network~~

~~d) Hybrid approach~~



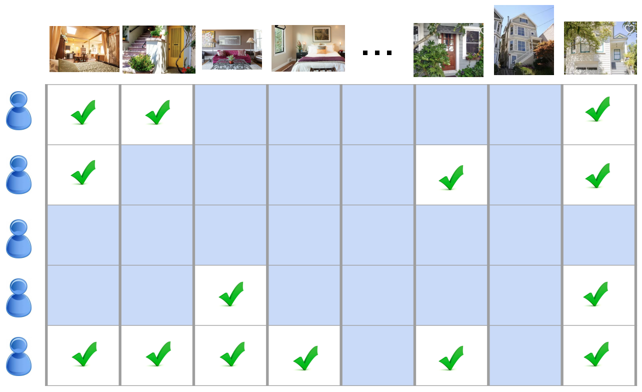
And, although it's possible that this recommendation came from a neural network or hybrid approach, it doesn't sound like it from the question.
The recommendation was made based on the similarity of features and content of the two apps. In fact, the features that relate the two apps and user are likely hand-engineered by some software engineer; they aren't learned solely through the data. We'll see more about how this works in the coming modules.



The user space and product space are **sparse** and **skewed**.

When you're developing any recommendation system, some common difficulties are helpful to keep in mind, and some care must be taken to address them.

For example, looking again at the user-item interaction matrix, note that the user space and the product space are sparse and skewed.

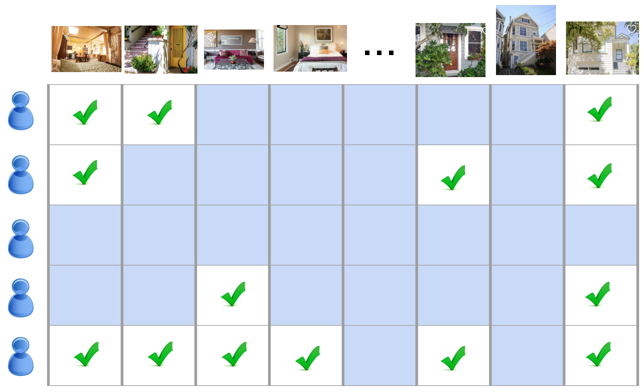


The user space and product space are **sparse**:

- Most items are rated by very few users.

We say the user-item matrix is sparse because there are potentially few interactions within the entire user-item space.
There could be billions of users and millions of items, and most of the entries in the matrix are zero.

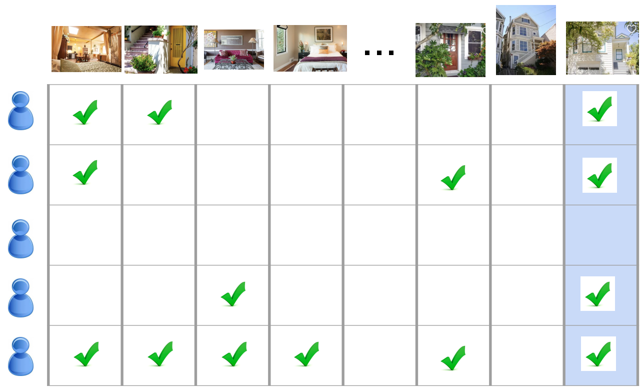
Sparse matrices are problematic because they take up a lot of memory and are slow to perform computations even though most of the computations are simply adding or multiplying by zero.



The user space and product space are **sparse**:

- Most items are rated by very few users.
- Most users rate only a small fraction of items.

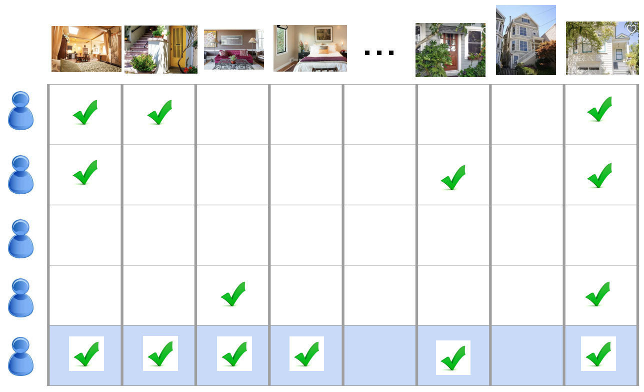
A typical person probably interacts with far less than 1% of your products.
On the other hand, most products are in the 'long-tail' of usage, and probably rated by less than 0.1% all users.



The user space and product space are **skewed**:

- Some properties are very popular.

In addition, the matrix is skewed!
Some properties might be very popular: maybe that property is a resort with a thousand cabins.

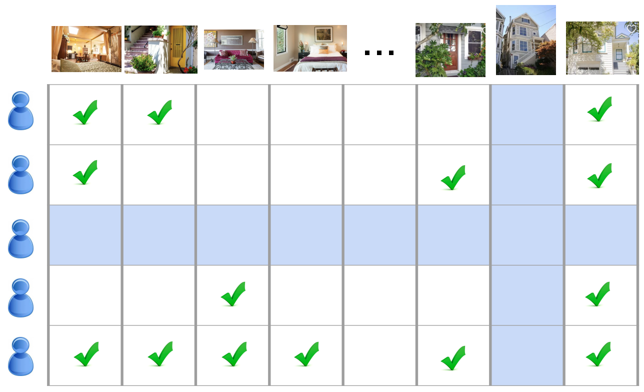


The user space and product space are **skewed**:

- Some properties are very popular.
- Some users are very prolific.

Or there could be a few users that are very prolific: maybe they are motorcycling around the country and staying in rental homes every day of the year. Or maybe some users just 'like' everything.

If you naively take **all** the ratings for **all** users, you risk overemphasizing certain users or products.

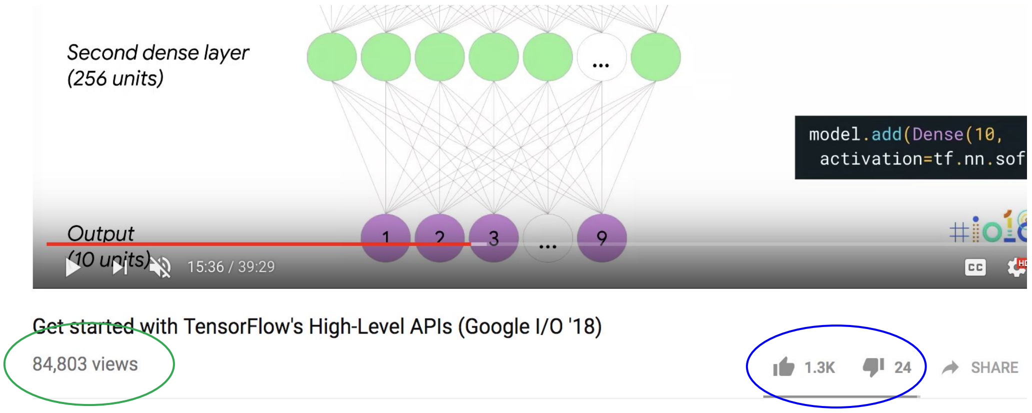


The **cold start problem** occurs when there aren't enough interactions for users or items.

In addition, a cold start can occur for both products and users. This happens when there aren't enough interactions or information for users or items.
For example, when a new user is added to a system, for a time the recommender has no past interactions with which to make new recommendations.

Or consider when a new item is added to a catalog: because collaborative filtering relies on user-item interactions, without this information, reliable recommendations for users aren't generated. In this situation, a content-based approach is better.

Explicit user feedback is often rare or unobservable



```
model.add(Dense(10,
 activation=tf.nn.sof
```

Get started with TensorFlow's High-Level APIs (Google I/O '18)

84,803 views

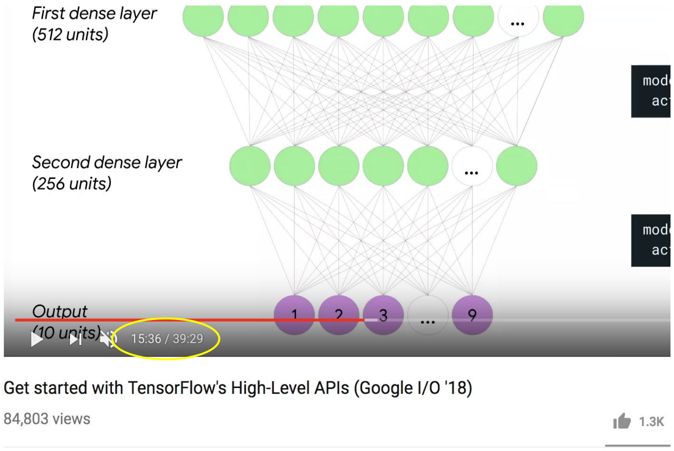
Another problem to keep in mind is the lack of explicit user feedback in the form of ratings or thumbs up or down.

Often we don't have explicit measures to feed to a recommender system. In situations like this, it is necessary to rely on implicit ratings.

For example, consider a YouTube video like this one. Of the 10s of thousands of views, there are substantially fewer 'up' votes or 'down' votes.

Implicit feedback is much more readily available

- # of clicks
- Play counts
- Fraction of video watched
- Site navigation
- Time spent on page



Get started with TensorFlow's High-Level APIs (Google I/O '18)

84,803 views

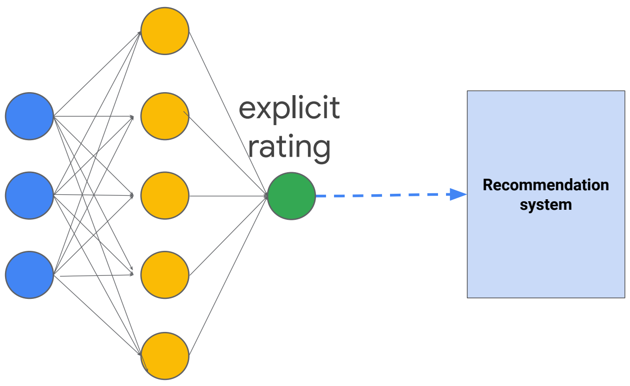
Instead, implicit measures must be used.

Things like
portion of video watched
number of clicks,
play counts,
or other information about the user interaction, like site navigation or time spent on the page.

In practical scenarios, implicit feedback like this is much more readily available, even though explicit user feedback is ground truth and preferred.

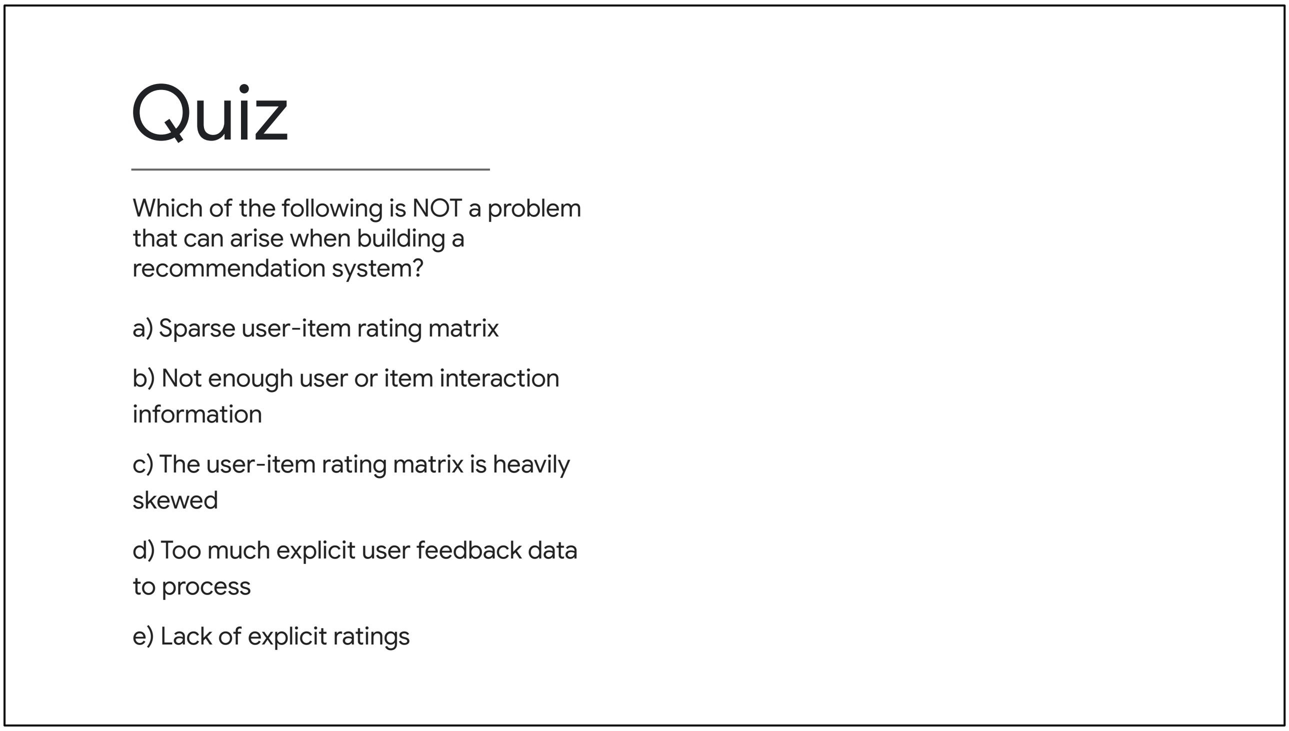
Implicit feedback is much more readily available

- # of clicks
- Play counts
- Fraction of video watched
- Site navigation
- Time spent on page



With enough data, you could train an initial model that uses implicit user feedback to learn an explicit rating.
Essentially, the input layer takes implicit ratings, and we use the explicit ratings as the label. This explicit rating is what we want to use as input when developing the recommendation engine.

At inference time, in the absence of an explicit rating, we use the trained initial model to infer an explicit rating that we can feed to the recommendation engine.
As we collect more data, these signal scores can be improved.



Quiz

Which of the following is NOT a problem that can arise when building a recommendation system?

a) Sparse user-item rating matrix

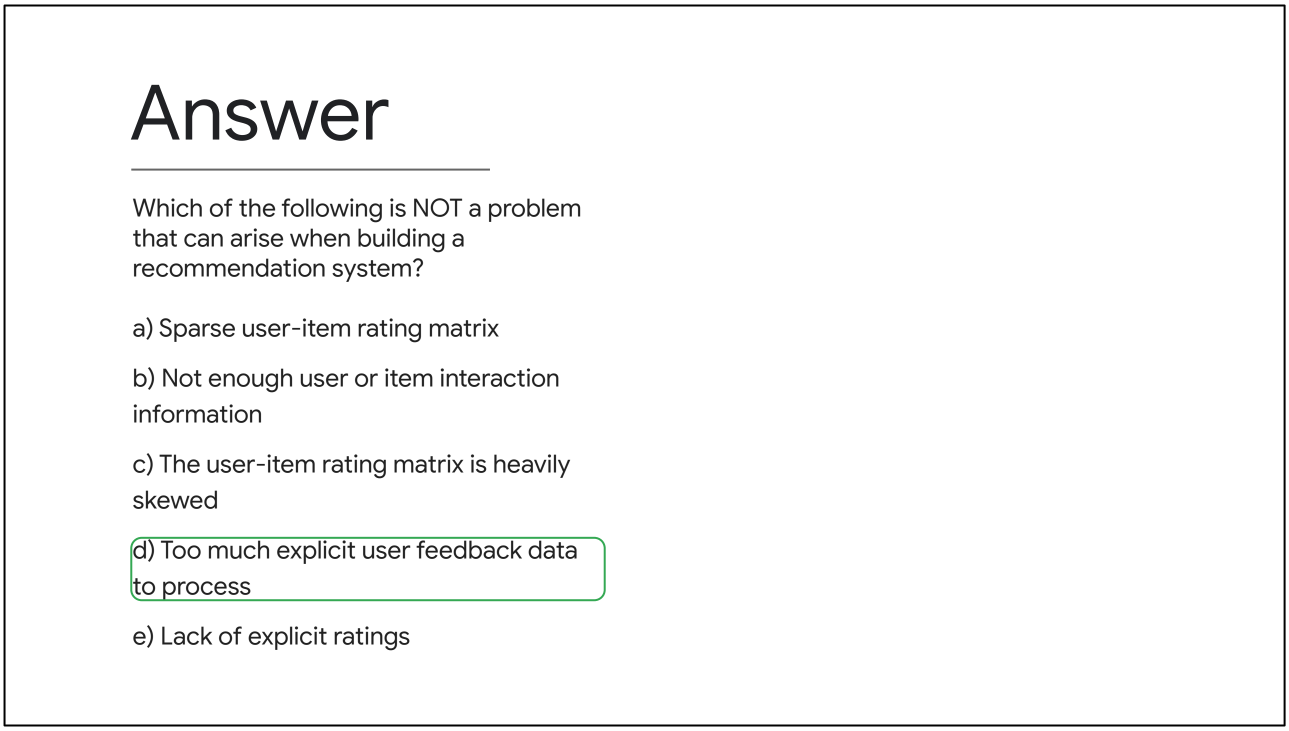
b) Not enough user or item interaction information

c) The user-item rating matrix is heavily skewed

d) Too much explicit user feedback data to process

e) Lack of explicit ratings

OK. Let's take a break for a quick quiz to check your understanding.
We've seen a couple issues that can arise when building recommendation engines.
Which of these is NOT a problem that can arise?



Answer

Which of the following is NOT a problem that can arise when building a recommendation system?

a) Sparse user-item rating matrix

b) Not enough user or item interaction information

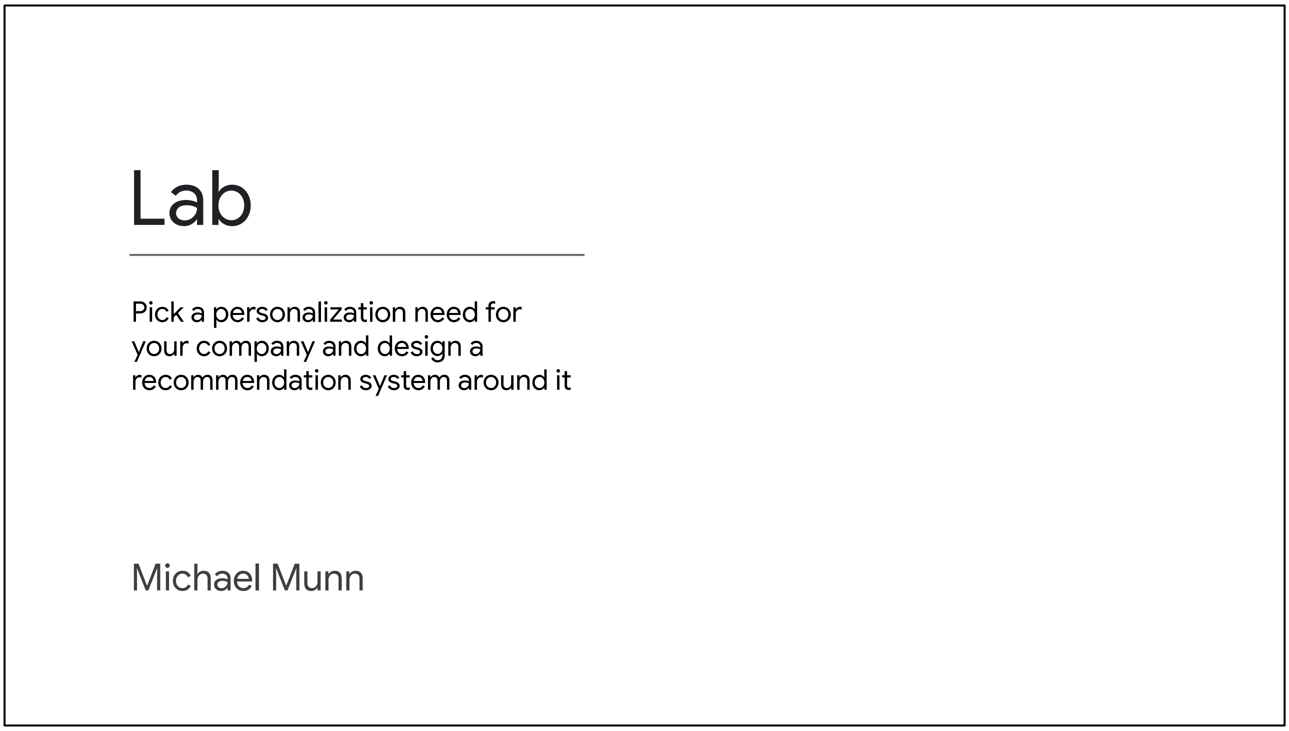
c) The user-item rating matrix is heavily skewed

[✓ d) Too much explicit user feedback data to process]

e) Lack of explicit ratings

That's right.

These are all problems that can arise when developing a recommender system except for d). It's unlikely that we'll have too much user rating data to use. In fact, it's more likely that we'll have a lack of these explicit user ratings to use at all.



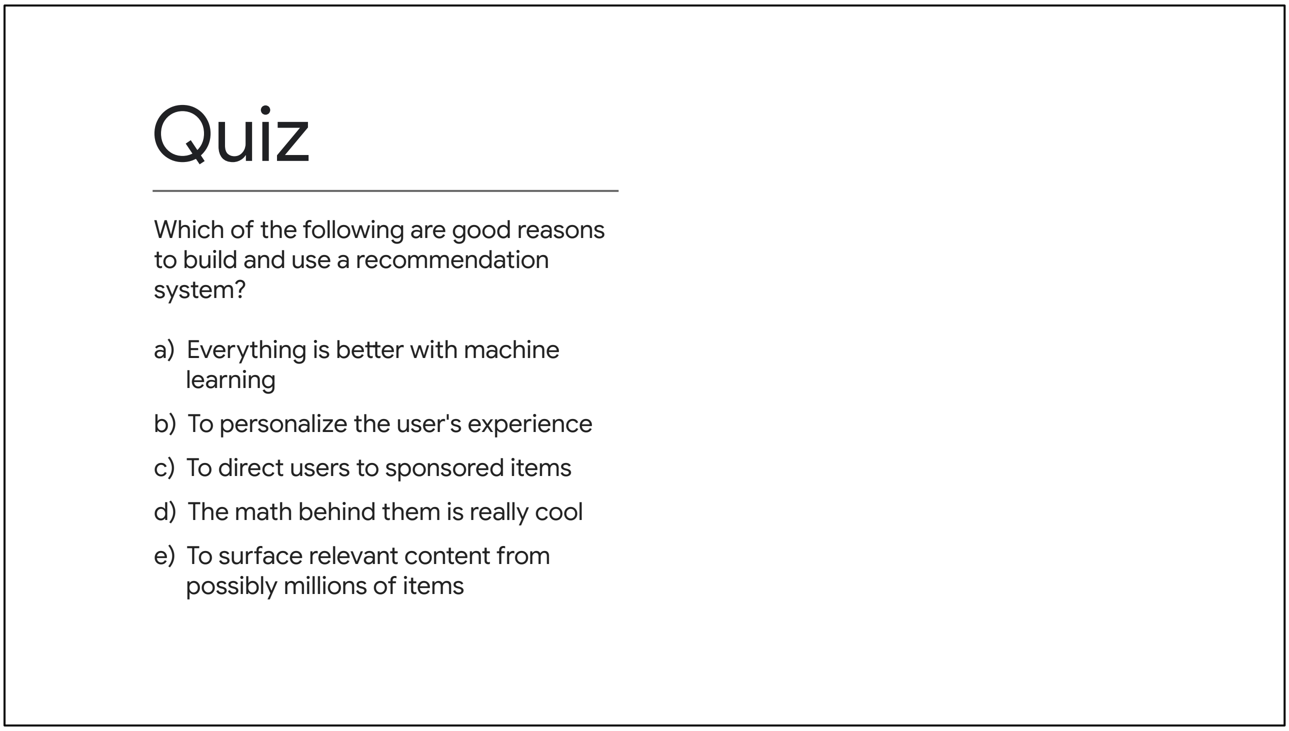
Lab

Pick a personalization need for your company and design a recommendation system around it

Michael Munn

Pick a personalization need for your company and design a recommendation system around it.

Think about what data you would need.
What are your features and targets?
How will you create content-based filter? Or a collaborative filter?
Do you have explicit ratings of user feedback you can use? Or will you have to rely on implicit feedback?
If so, what implicit feedback is available?
How will you address the cold-start problem? Could you benefit from a hybrid approach?



Quiz

Which of the following are good reasons to build and use a recommendation system?

a) Everything is better with machine learning

b) To personalize the user's experience

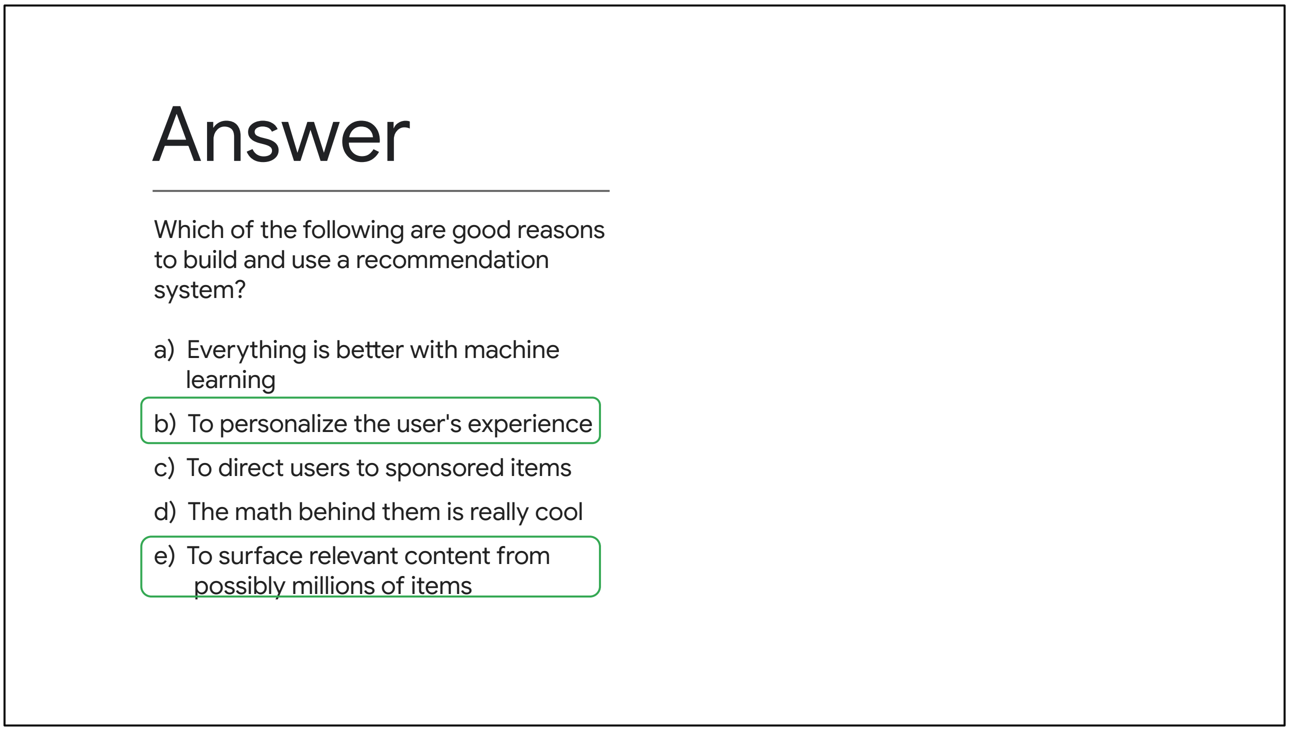
c) To direct users to sponsored items

d) The math behind them is really cool

e) To surface relevant content from possibly millions of items

Let's end this module with a quiz.

Which of the following are good reasons to build and use a recommendation engine? There could be more than one right answer.



Answer

Which of the following are good reasons to build and use a recommendation system?

a) Everything is better with machine learning

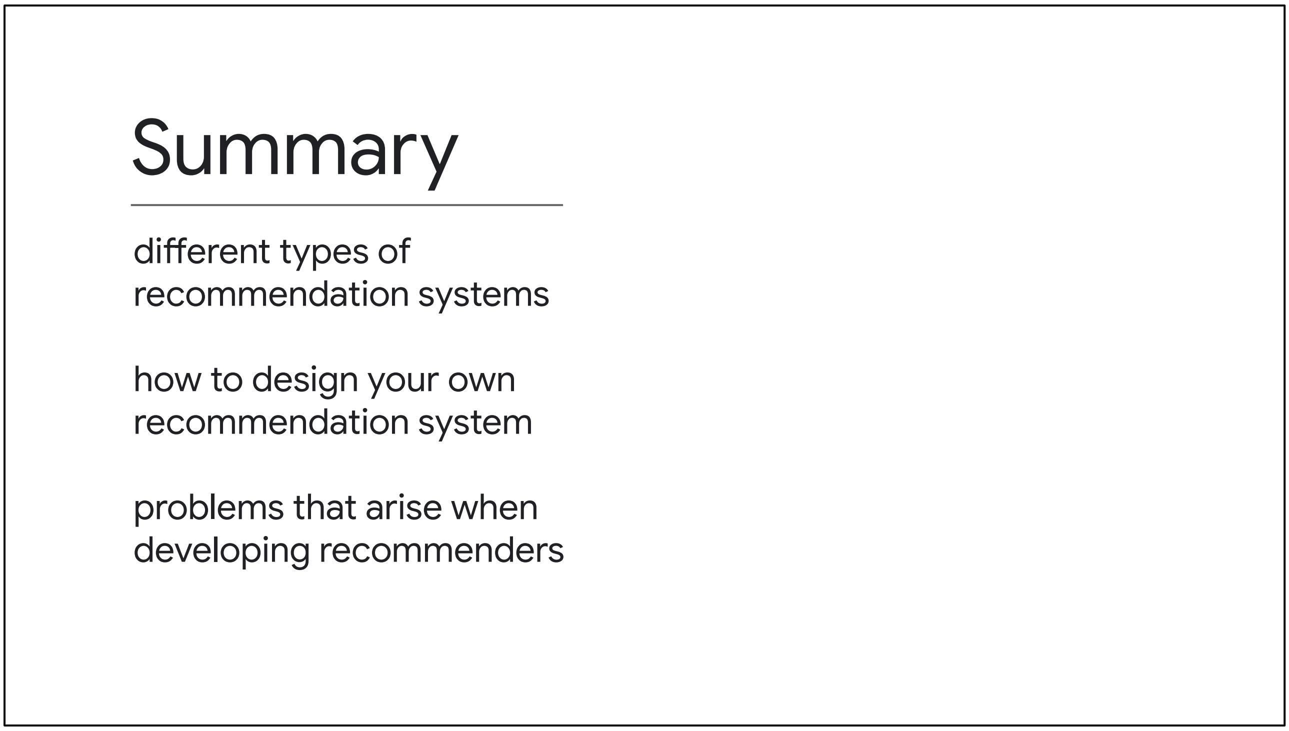
[✓ b) To personalize the user's experience]

c) To direct users to sponsored items

d) The math behind them is really cool

[✓ e) To surface relevant content from possibly millions of items]

That's right.
Even though the math behind recommender systems is really cool, that is not a good enough reason alone to develop one.
Some good reasons are that they personalize the user's experience, or that recommenders provide a way to surface relevant content within millions of items, like recommending YouTube videos you might never have found yourself.



Summary

different types of recommendation systems

how to design your own recommendation system

problems that arise when developing recommenders

In summary, we've had a brief introduction to the different types of commonly used recommender systems.
We've discussed the difference between content-based and collaborative filtering. We've seen how a knowledge based approach can be used. And we've seen how deep neural networks could be applied in this setting as well.

We’ve discussed how to design and build your own recommendation engine. And we've seen some of the problems that can arise, like the cold start problem or lack of explicit user feedback. We've also discussed ways in which these problems can be addressed.

In the next modules we'll go deeper into the details of how these recommender systems work and even see how to build them in Tensorflow.