

# Recommender Systems



# TABLE OF CONTENTS

**01**

**Introduction**

**03**

**Content-Based  
Filtering**

**05**

**Hybrid  
Recommenders**

**Knowledge-Based  
Filtering**

**Collaborative  
Filtering**

**02**

**04**

**01**

A thick, orange, hand-drawn brushstroke that curves under the number '01', starting from the left, looping around the bottom, and ending on the right.

# **Introduction**

# Definition

**Recommender systems** are a type of information filtering system that recommend items to users based on their past behavior, preferences, or other relevant factors.

The goal of a recommender system is to suggest items that the user is likely to be interested in, thereby improving their experience and increasing their engagement with the system.

Which **problem** do Recommender Systems **solve** ?

*Recommender Systems were developed to address the problem of information overload, which occurs when users are presented with a large amount of information and have difficulty finding the most relevant or interesting items.*

?

# Users & Items

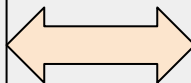
## Users



A **user** refers to an individual who interacts with the system and receives recommendations for items based on their past behavior, preferences, and other relevant factors.

There are two types of users:

- **Active Users** - they have generated data through their interactions with the system.
- **Passive Users** - they do not actively interact with the system, but their data may still be used to generate recommendations.



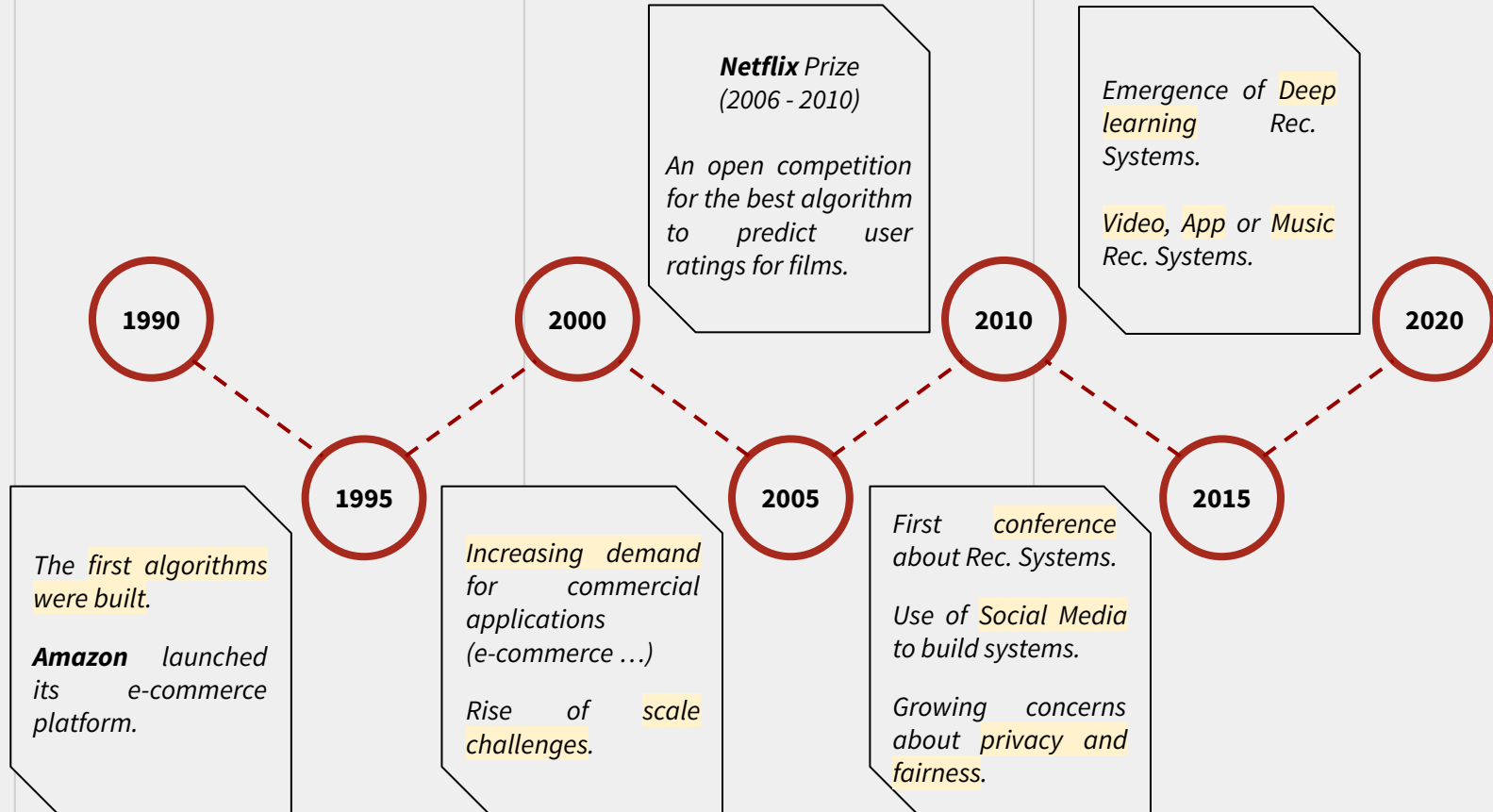
## Items

An **item** refers to a product, service, piece of content, or any other entity that can be recommended to a user such as products, movies, songs, articles, posts and so on.

Items can be represented in different ways, depending on the type of recommender system and the data available.

*For example, items on a website may be represented by their product name, category, and price while items on a music streaming service may be represented by their title, artist, genre, and album.*

# History



# Applications

**Recommender systems** have a wide range of applications in different domains that is constantly expanding as new data and technologies become available.



**Product recommendations** on e-commerce websites  
**Personalized marketing campaigns**  
**Customer segmentation and targeting**

**Friend or follower recommendations** on social media platforms  
**News feed recommendations** based on user interests and preferences  
**Content recommendations** on video sharing platforms



**Movie recommendations** on streaming platforms  
**Music recommendations** on streaming platforms  
**Book recommendations** on e-commerce websites

# Applications (2)

**Recommender systems** have a wide range of applications in different domains that is constantly expanding as new data and technologies become available.

***Treatment recommendations** based on patient data and medical history*

***Clinical decision** support systems for healthcare providers*

***Drug recommendations** and adverse event prediction systems*



***Article recommendations** on news websites*

***Content recommendations** on online learning platforms*

***Research paper recommendations** on academic websites*

***Course recommendations** based on academic records and interests*

***Textbook recommendations** based on course curriculum and student performance*

***Personalized learning paths** based on learning styles and abilities*






# Amazon

**Amazon's** recommender system analyzes user data such as purchase history, search queries, product ratings, and other interactions with the website to build a profile of each user's preferences and interests.

The system then uses this information to recommend products that are likely to be of interest to the user.



**Recommender Systems: The Textbook** 1st ed. 2016 Edition  
by Charu C. Aggarwal (Author)  
★★★★★ 56 ratings

1 Highlights found by Fakespot


See all formats and editions

eTextbook	Hardcover	Paperback
\$21.00 - \$45.02	<del>\$46.00</del> - <del>\$46.14</del>	<del>\$50.03</del> - <del>\$69.99</del>
Read with Our <b>Free App</b>	14 Used from \$34.99 13 New from \$46.14	6 Used from \$50.03 10 New from \$65.96


This book comprehensively covers the topic of recommender systems, which provide personalized recommendations of products or services to users based on their previous searches or purchases. Recommender system methods have been adapted to diverse applications including query log mining, social networking, news recommendations, and computational advertising. This book synthesizes both fundamental and advanced topics of a research area that has now reached maturity. The chapters of this book are organized into three categories:

Read more


Customers who viewed this item also viewed




**Practical Recommender Systems**  
> Kim Falk  
★★★★★ 35  
Paperback  
\$49.99  
\$12.77 shipping




**Recommendation Engines (The MIT Press Essential Knowledge series)**  
> Michael Schrage  
★★★★★ 70  
Paperback  
\$11.99  
\$10.55 shipping




**Machine Learning Design Patterns: Solutions to Common Challenges in Data Preparation, Model Training, and Evaluation**  
> Vallappa Lakshmanan  
★★★★★ 242  
Paperback  
\$36.99  
\$12.59 shipping



**Statistical Methods for Recommender Systems**  
Deepak K. Agarwal  
★★★★★ 12  
Hardcover  
\$49.89  
\$13.93 shipping  
Only 6 left in stock (more ...)



**Designing Machine Learning Systems: An Iterative Process for Building Scalable and Robust Machine Learning Applications**  
> Chip Huyen  
★★★★★ 109  
Paperback  
**#1 Best Seller** in Business Intelligence Tools  
\$43.49  
\$12.31 shipping



**Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Applications**  
> Aurélien Géron  
★★★★★ 3,267  
Paperback  
**#1 Best Seller** in Computer Vision & Pattern Recognition  
\$56.16  
\$15.28 shipping

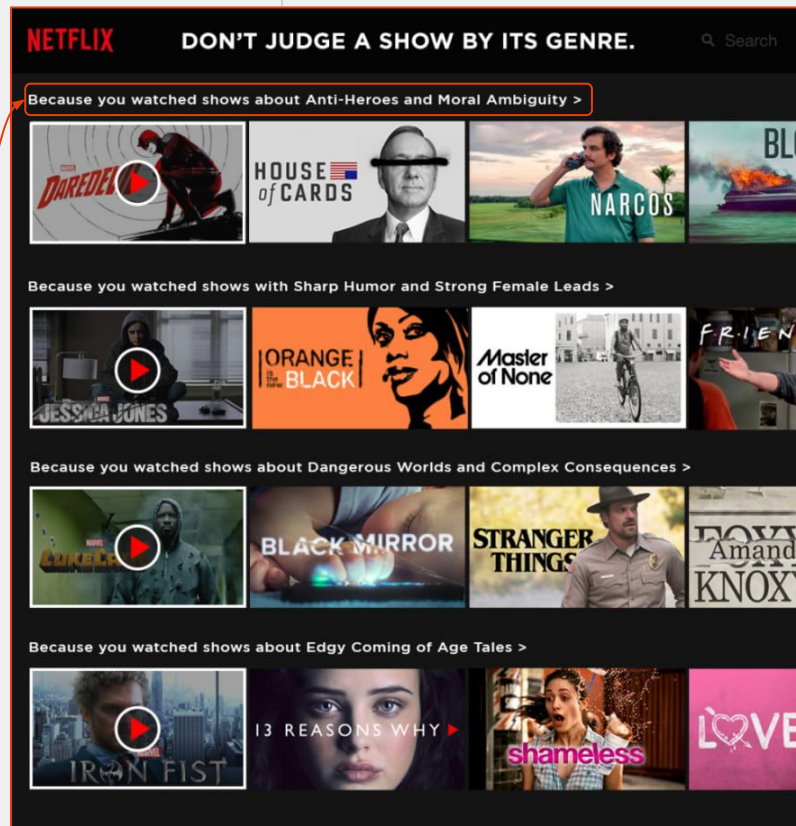
Page 1 of 10

# Netflix

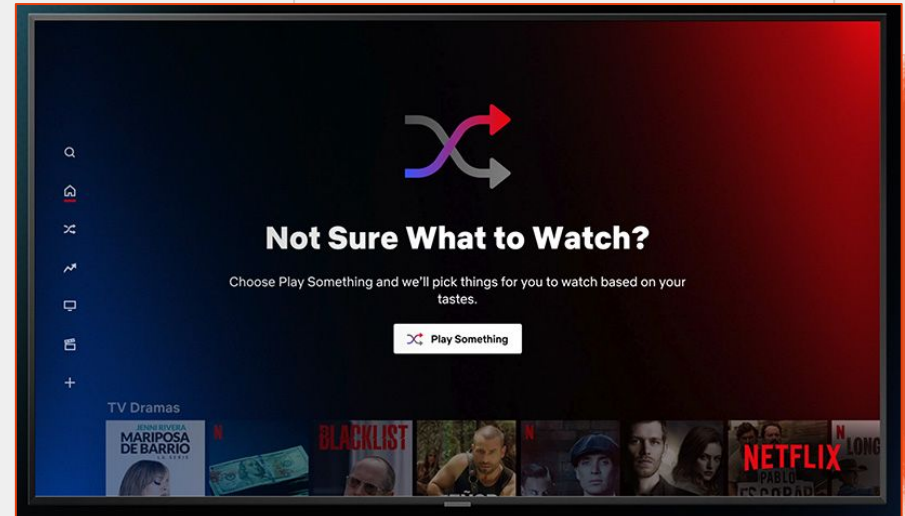
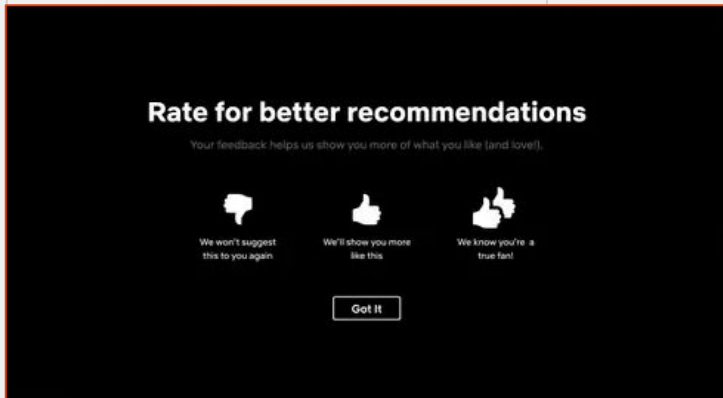
**Netflix** has been using recommender systems to provide personalized movie and TV show recommendations to its users since its inception.

It analyzes user data such as viewing history, ratings, and search queries to build a model of each user's preferences and interests.

The system then uses this information to recommend movies and TV shows that are likely to be of interest to the user.



# Netflix (2)



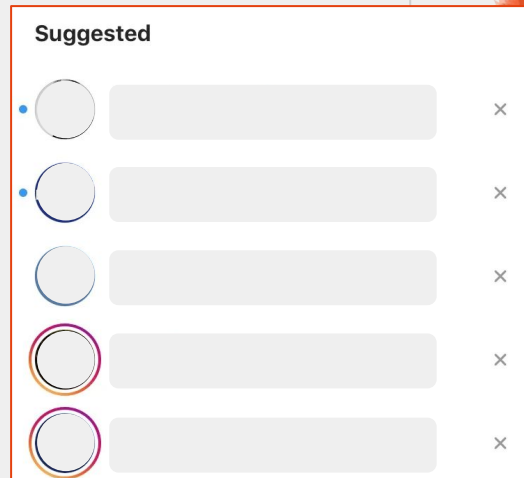
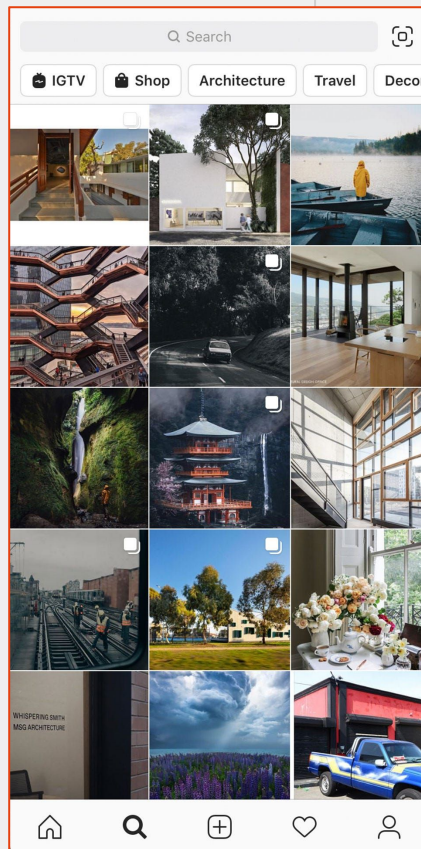
*All of these small interactions will be taken into account to build personalized recommendations.*

# Instagram

**Instagram** has been using recommender systems to personalize content recommendations for its users.

The recommender system at Instagram analyzes user data such as past engagement with content, demographic information, and user behavior to build a profile of each user's interests and preferences.

The system then uses this information to recommend new content, including posts from other users, videos, and ads, that are likely to be of interest to the user.



# The power of Recommender Systems

amazon

According to some sources,  
more than **30% of its sales**  
are done thanks to the  
platform Recommender  
Systems.

NETFLIX

According to some sources,  
more than **70% of the**  
**movies watched** could  
come from the platform  
Recommender Systems.



**All the content** on  
Instagram is recommended.

# Other examples



## Spotify

Recommendations based on your music tastes and listenings



## LinkedIn

"You may also know" or "You may also like" type of recommendations



## Tinder

The entire application is based on the recommendation of other users



# Data

**Recommender systems** rely on data that is specific to each user and item. The data used in recommender systems can be broadly categorized into two types:

- **Explicit Data** - *it refers to data that is explicitly provided by the user, such as ratings, reviews, and feedback.*
- **Implicit Data** - *it refers to data that is collected implicitly from user behavior, such as clickstream data, search queries, and purchase history.*

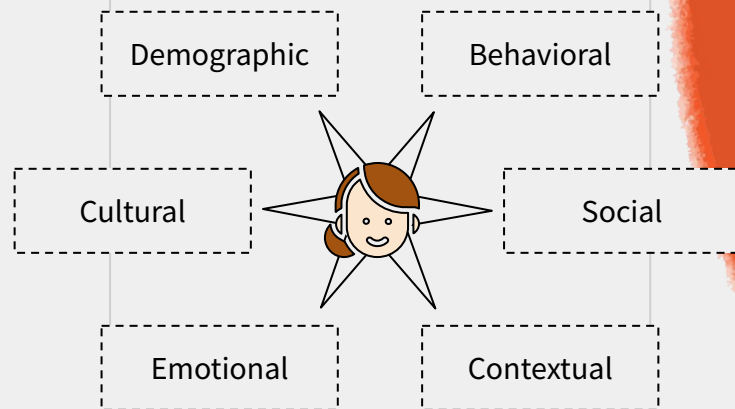
**Quantity** and **quality** of the data are important factors in the performance of recommender systems. Systems with larger and more diverse datasets tend to perform better, as they are able to capture a wider range of preferences and interests.

# Datasets

Explicit and implicit data are generally used to generate three type of datasets:

- The **User detail** dataset - *it contains information about the users who interact with the system like demographic data, such as age, gender, location, and occupation, as well as behavioral data, such as purchase history, search queries, and ratings. User detail datasets are used to build user models, which are used to generate personalized recommendations for each user.*

	Age	Location	...	Number of Friends	Personality
User 1	25	US	...	84	Creative
User 2	29	India		101	Empathic
User 3	19	China		74	Mindful
User 4	42	Russia		53	Ambitious



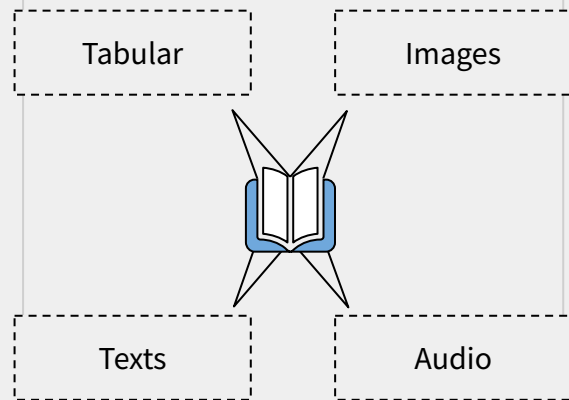


# Datasets (2)

Explicit and implicit data are generally used to generate three type of datasets:

- The **Item detail** dataset - *it contains information about the items that are available in the system like attributes such as the title, genre, artist, author, and release date. These datasets are used to build item models, which are used to identify patterns and similarities between different items, and to generate recommendations.*

	Product Name	Price (€)	...	Description	Color
Item 1	Table	135	...	Item description	Brown
Item 2	Game	80		Item description	N/A
Item 3	Sweatshirt	39		Item description	Navy
Item 4	Phone	450		Item description	White



# Datasets (3)

Explicit and implicit data are generally used to generate three type of datasets:

- The **User-Item interaction** dataset - *it contains information about the interactions between users and items in the system such as ratings, reviews, clicks, purchases, and other user actions. User-item interaction datasets are used to train and evaluate recommendation algorithms, by providing a set of known user-item interactions that can be used to test the relevance of recommendations.*

	User ID	Item ID	Ratings	Reviews	...
Interaction 1	5	14	3	Item review	...
Interaction 2	12	18	N/A	N/A	
Interaction 3	48	151	5	Item review	
Interaction 4	27	86	4	Item review	

*The User ID and Item ID columns are mandatory as they allow to map the interaction between the users and the items*

# Interactions

**Interactions** are the primary means by which a recommendation system learns about a user's preferences. They can be classified into two categories:

## Explicit



**Explicit** interactions are **direct** inputs from the user that reflect their preferences.

Some common ones include:

- Ratings
- Reviews
- Likes/Dislikes
- Favorites



## Implicit

**Implicit** interactions are **inferred** from a user's behavior and are not direct indications of preference.

Some common ones include:

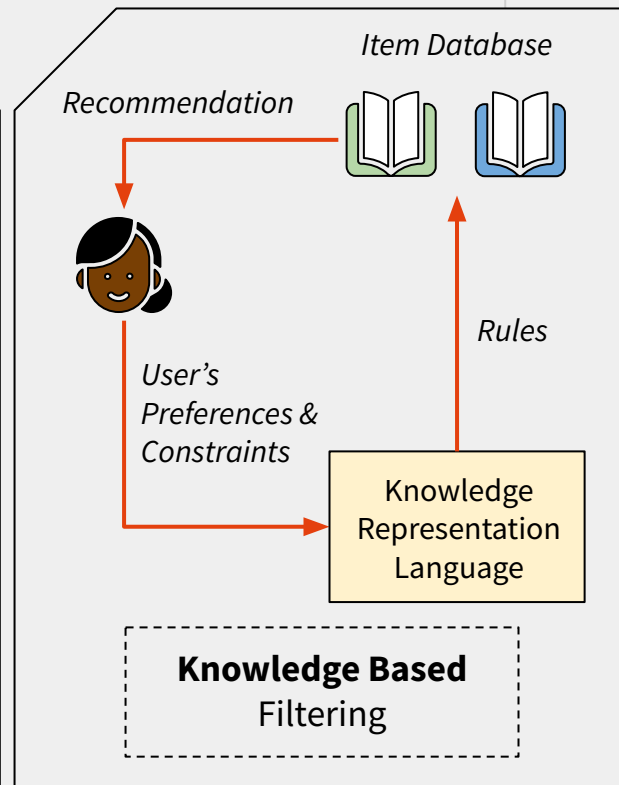
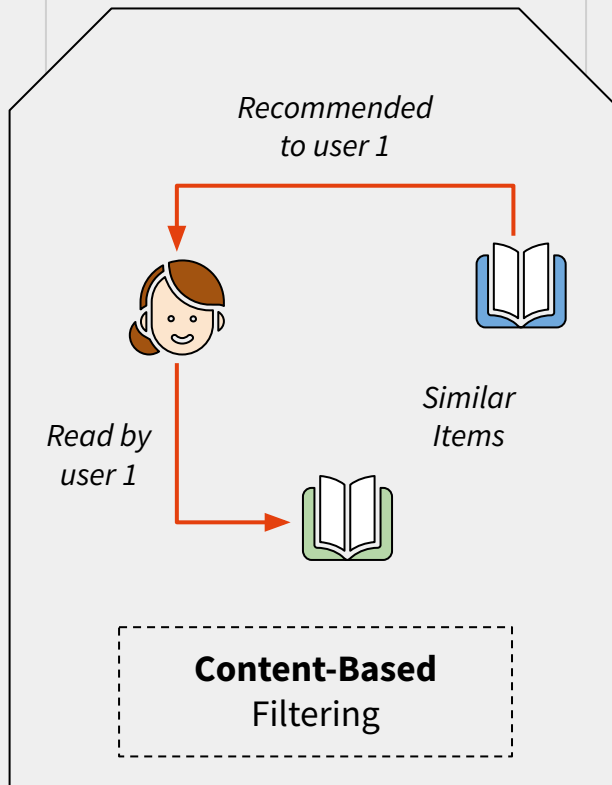
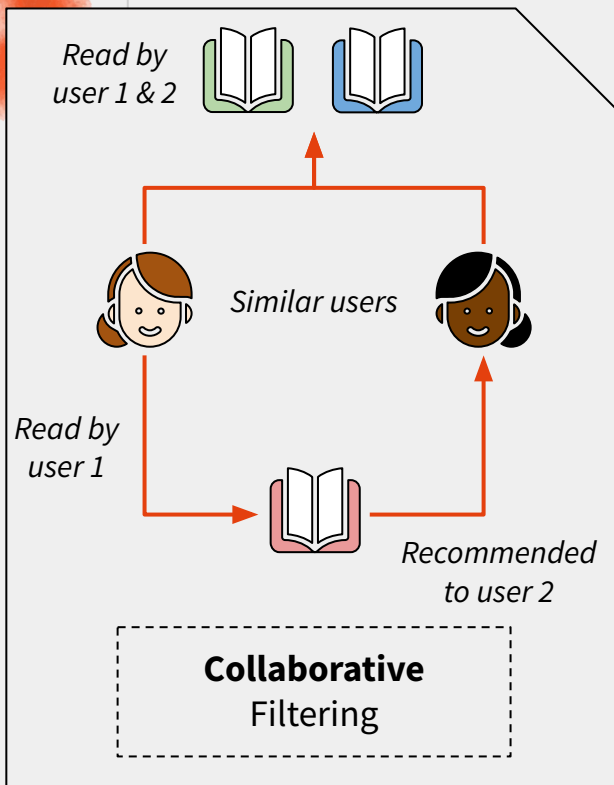
- Clicks
- Purchase history
- Viewing history
- Time Spent
- Navigation patterns

# Algorithms

**Recommender systems** techniques can be divided in four main categories:

- **Collaborative filtering** - *it generates recommendations by using the past behavior of users and their interactions with items to identify similar users and recommend items that they have liked or interacted with.*
- **Content-based filtering** - *it generates recommendations by analyzing item attributes and features, building a profile of the user's preferences based on past interactions, and recommending other items with similar attributes.*
- **Knowledge-based recommendation** - *it generates recommendations by representing user preferences using a knowledge representation language, matching them to items that satisfy the criteria and utilizing domain-specific knowledge to produce relevant recommendations.*
- **Hybrid recommendation** - *it combines the aforementioned techniques to generate recommendations that are more accurate and relevant by leveraging the strengths of each technique.*

# Algorithms (2)



# Recommendation

**Recommender systems** can suggest new items to the user. Many systems are designed with a two-step process composed of a retrieval step followed by a ranking step. This is especially common in systems that need to choose from an extremely large pool of items, such as video or product recommendation systems.

1. **Retrieval** - The goal at this stage is to cast a wide net and retrieve any items that the user might possibly be interested in. This is typically a coarse-grained step designed for high recall, which means it aims to identify as many relevant items as possible, even if it includes some irrelevant ones.
2. **Ranking** - The goal at this stage is to fine-tune the recommendations and present the most relevant items at the top of the list. This is typically a fine-grained step designed for high precision, which means it aims to ensure that the top-ranked items are highly relevant.

# Metrics

**Recommender systems** can be evaluated in two ways:

- Evaluating the proportion of recommended items that are relevant - *Precision, Recall, F1 score, Mean Average Precision at k (MAP@k), Normalized Discounted Cumulative Gain (NDCG), Hit Rate, Coverage, Diversity, Novelty ...*
- Evaluating the ratings of the recommended items - *Root Mean Squared Error (RMSE) ...*

While these metrics can provide a quantitative measure of performance, it's also important to consider qualitative evaluation methods, like user studies or A/B testing. These can provide a more holistic view of the system's performance, including factors like user satisfaction that are hard to quantify.

# Metrics (2)

**Precision** measures the proportion of recommended items that are relevant while **recall** measures the proportion of relevant items that are recommended. These are the most basic metrics to evaluate a recommender systems. Often, a trade-off between precision and recall is needed and the **F1 score** can be used to balance them.

The **Mean Average Precision at k** works as follow:

- **Precision at k** ( $P@k$ )- *it calculates the proportion of recommended items in the top-k that are relevant for the user.*
- **Average Precision at k** ( $AP@k$ ) - *it is an extension of  $P@k$  that takes into account the order of the recommendations. It's the average of  $P@k$  values for each k up to the current k.*
- **Mean Average Precision at k** ( $MAP@k$ ) - *it is the average of  $AP@k$  values for all users.*

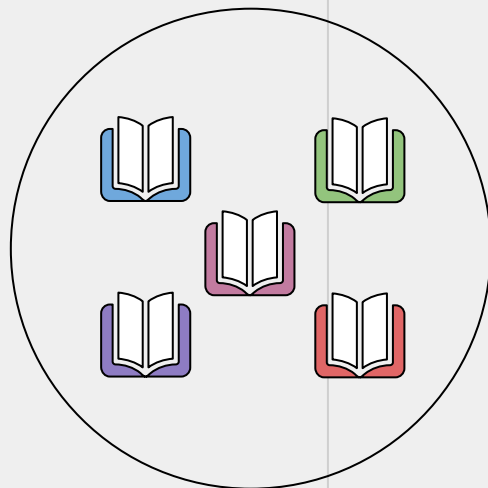


# Metrics (3)

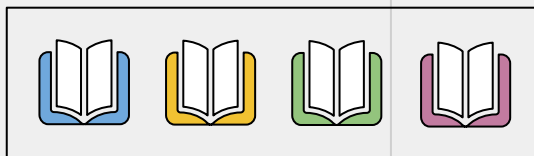


**User's relevant items**

*Not to be confused with the user's profile !!*



**Recommendations**



Top

1

2

3

4

## **P@1 - Precision at 1**

*The blue book is in the user's relevant items.*

$$P@1 = 100\%$$

## **P@2 - Precision at 2**

*The yellow book is **not** in the user's relevant items.*

$$P@2 = 1/2 = 50\%$$

## **P@3 - Precision at 3**

*The green book is in the user's relevant items.*

$$P@3 = 2/3 = 66\%$$

## **P@4 - Precision at 4**

*The pink book is in the user's relevant items.*

$$P@4 = 3/4 = 75\%$$

## **AP@4 - Average Precision at 4**

$$\begin{aligned} AP@4 &= (P@1 + P@2 + P@3 + P@4) / 4 \\ &= (1 + 0.5 + 2/3 + 0.75) / 4 \\ &= 72.75\% \end{aligned}$$

# Metrics (4)

**Coverage**, **Diversity** and **Novelty** do not focus on positive/negative samples like the Precision, Recall and MAP@k, but are crucial for understanding the overall quality of the recommendations.

## Coverage

It is a measure of how many items the recommender system is able to recommend out of all possible items in the catalog.

High coverage can be beneficial in scenarios where users have diverse interests or there are many niche items.

However, it should not come at the expense of item relevance.

## Diversity

It is a measure of how different the recommended items are from each other.

Increasing diversity is beneficial in scenarios where users are looking for something new or different but can often come at the cost of less relevant items.

## Novelty

It is a measure of how new the recommendations are.

Increasing novelty can help users discover items they may not have found otherwise, which can improve user satisfaction and engagement.

However, increasing novelty can often come at cost of less relevant items.



02

# **Knowledge-Based Filtering**

# Definition

**Knowledge-based filtering** is a type of recommendation approach where the system suggests items to the user based on explicit knowledge about the items and the user's requirements.

Such explicit knowledge can include details about the item's characteristics or how the item meets a particular user need or preference and it usually involves an interactive process with the user(s).

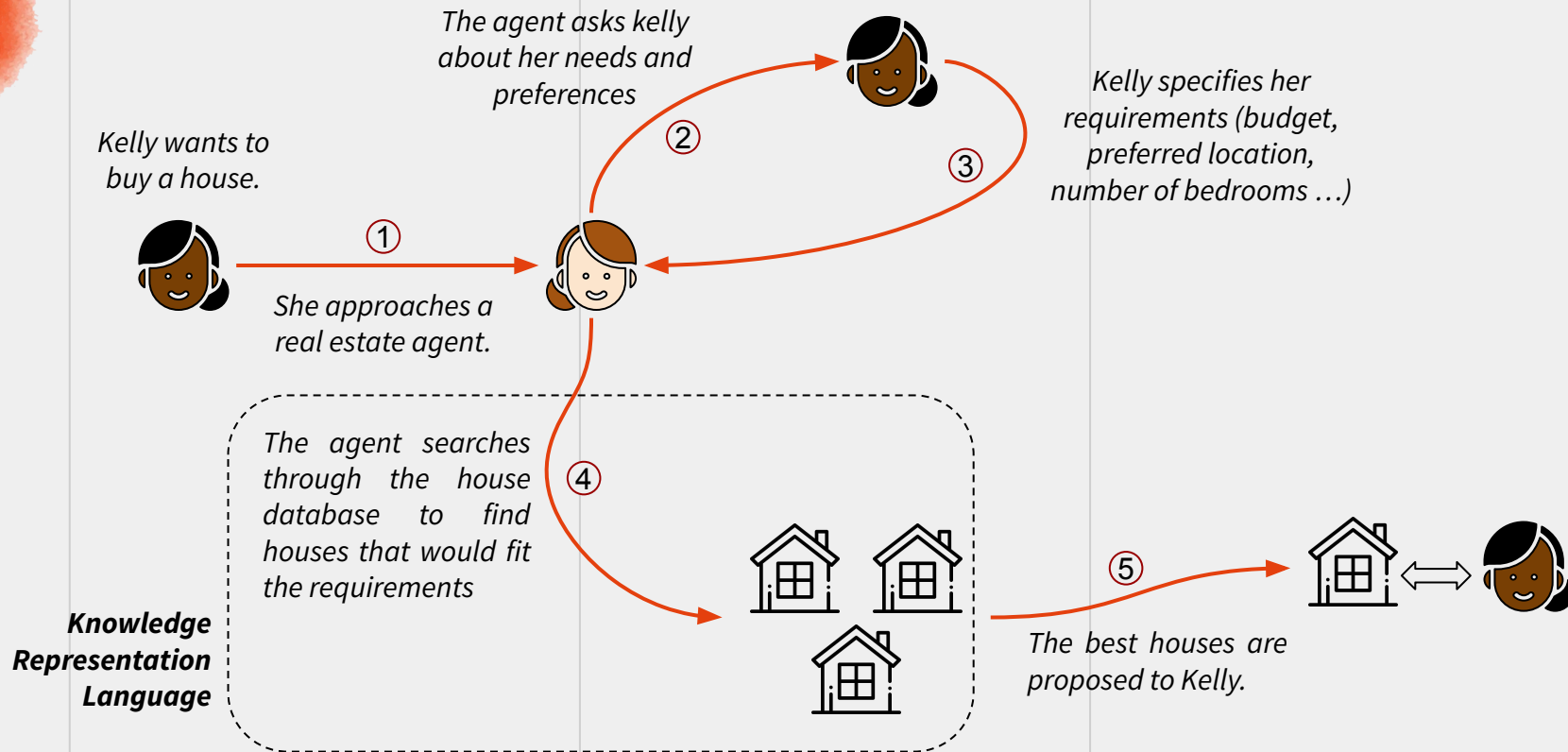
Which **problem** do knowledge-based systems **solve** ?

?

*Knowledge-based systems were developed for situations where there's little past interaction data available, or where user requirements are very specific and unique. This problem is called the cold start problem\*.*

\* We will talk about this problem later in the Collaborative Filtering part

# Example



# Knowledge Representation Language

**Knowledge Representation Language** is a form of structured language used to define and represent knowledge about items and user requirements. It is used by the recommendation system to understand the properties of items, user needs, and the rules or relationships that dictate which items would satisfy which user requirements.

These languages generally need to be structured and consistent so that the recommendation system can understand and use the represented knowledge effectively. They could be based on simple attribute-value pairs, logical rules and constructs, or more complex structures.



## ***Attribute-value pairs***

*Title: Blade Runner  
Author: Philip K. Dick  
Genre: Fiction  
Publication Year: 1968*

## ***Logical Rule***

*If the user's preferred genre is Fiction, and the user prefers books published after 1950, and the book's genre is Fiction, and the book's publication year is after 1950, then recommend the book.*

# Knowledge Update

Knowledge-based systems need to be updated by incorporating new information about items and users. This knowledge can be collected using different methods:



## Direct input from users

*They can provide explicit feedback on items they use or experience (ratings, reviews, answers ...)*



## Expert input

*They can provide valuable information about the attributes and characteristics of items. They can also provide rules or criteria that guide the recommendation process.*



## Automated data collection

*(web scraping, data mining)*

*It can be used to collect large amounts of information about items from various sources.*



## Integration with databases or APIs

*It can be used to access detailed information about items.*

# Recommendation

There are mainly two approaches for generating recommendations in a knowledge-based system:

## *Case-Based Reasoning*

Recommendations are generated by finding past cases similar to the current one. Each case consists of a problem and its solution.

The **challenge** here is defining what similar means and how to measure it, which can be achieved through different distance measures.

## *Rule-Based Reasoning*

Recommendations are generated by a set of predefined rules that are usually in the form of if-then statements. They capture the relationship between the item's attributes and the user's needs.

The **challenge** here is defining the rules, which often requires expert knowledge. Additionally, rules generally need to be maintained over time.



**03**

# **Content-Based Filtering**

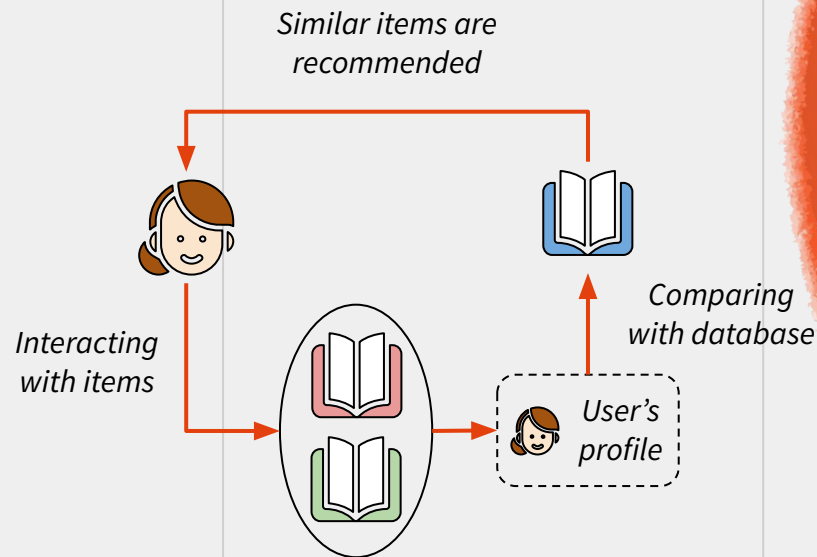
# Definition

**Content-based filtering** is a type of recommendation approach where the system makes suggestions based on the individual user's behavior and the features of items.

The system learns the user's preference thanks to the features of items the user has interacted with, and uses these preferences to predict the user's interest in other items.

Content-based filtering is a three-steps recommendation system:

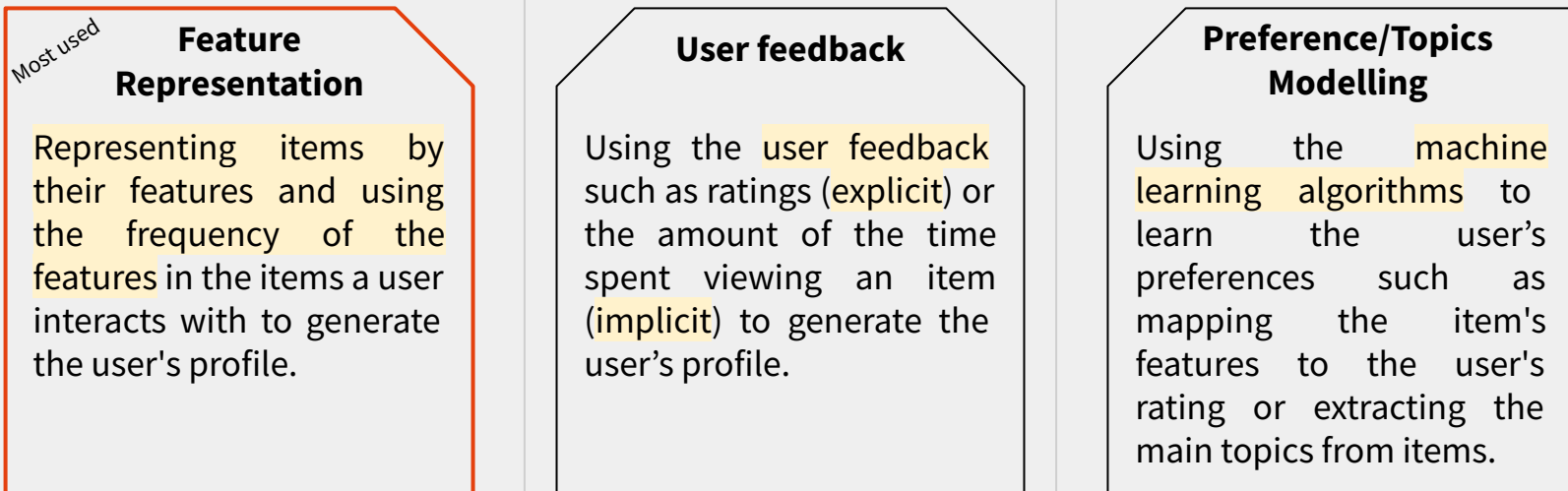
1. Building user's profile by analyzing the user's history of interactions
2. Measuring similarity between the user's profile and the features of other items.
3. Recommending the most similar items to the user's profile



# User's Profile

Building the **user's profile** or preferences involves creating a representation of the user's interests based on their interactions with items. These interactions can be explicit, such as ratings or reviews, or implicit, such as viewing or purchasing an item.

A very simple method would be to use the last item interacted with but there are also several more complex, and better, methods to build the user's profile in content-based filtering systems:



# Similarity Measure

After the user's profile is built, the system can then measure the **similarity** between the user's profile and the features of other items. The goal is to identify items that are most similar to the user's preferences. The goal is to identify items that are most similar to the user's preferences.

However, this step should only be done along with the user's profile obtained with a feature representation method.

## What's a *similarity* measure ?

*A similarity measure is a way to quantify the similarity or dissimilarity between two objects, entities, or data points. It is generally a mathematical measure that determines how close two items are in terms of their characteristics, attributes, or features.*

?

# Similarity Measure (2)

There are many different similarity measures, some of the most common ones include:

## Pearson Correlation

It calculates the correlation between two vectors, which in this case would be the user's profile vector and the item feature vector.

It ranges from -1 (*perfect negative correlation*) to 1 (*perfect positive correlation*), with values close to 0 indicating no correlation.

$$\rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$

## Euclidean Distance

It calculates the distance between two points, which in this case would be the user's profile and the item's features.

The smaller the Euclidean distance, the more similar the user and item vectors are.

$$d_{X,Y} = \sqrt{\sum_i^n (X_i - Y_i)^2}$$

## Cosine Similarity

Most used

It measures the cosine of the angle between two vectors, which in this case would be the user's profile vector and the item feature vector.

A cosine value of 1 indicates that the vectors are identical, a value of 0 means no similarity, and a value of -1 means they are diametrically opposed.

$$Sc_{X,Y} = \frac{\sum_i^n X_i Y_i}{\sum_i^n X_i^2 \sum_i^n Y_i^2}$$

# Recommendation

In **Content-based filtering**, recommendation are made by choosing the most similar item(s) to the user's profile.



User's profile

Fiction	Adventure	Comedy
0.33	0.66	0.75

*In the above example, 33% of the user seen movies were fictions, 66% were adventures and 75% were comedies.*



Item's features

	Fiction	Adventure	Comedy
Movie n°1	0	0	1
Movie n°2	0	1	1
Movie n°3	1	0	1
Movie n°4	1	1	0

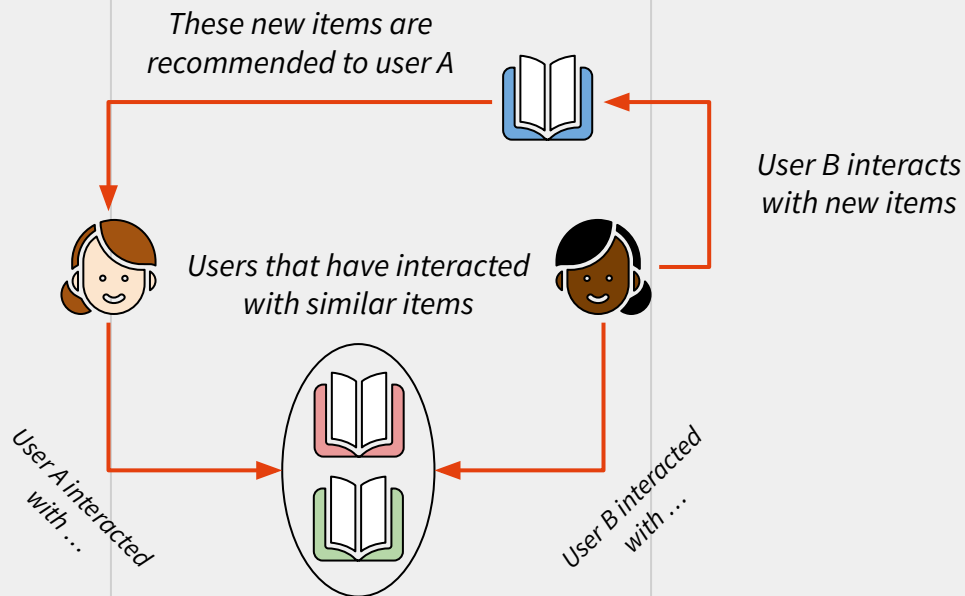


# **Collaborative Filtering**

# Definition

**Collaborative Filtering** is a method used in recommender systems that predicts a user's interests by collecting preferences from many users.

The fundamental assumption is that if a user A has the same opinion as a user B, A is more likely to have B's opinion on a different issue.





# Memory vs Model

**Collaborative Filtering** methods can be divided into two groups:

## Memory-Based



A type of collaborative filtering that directly uses the entire user-item database to generate a prediction. It involves techniques like **user-based** and **item-based** collaborative filtering that calculate recommendations based on the similarity between users or items.

It is straightforward to implement, but it often suffers from scalability and sparsity issues.



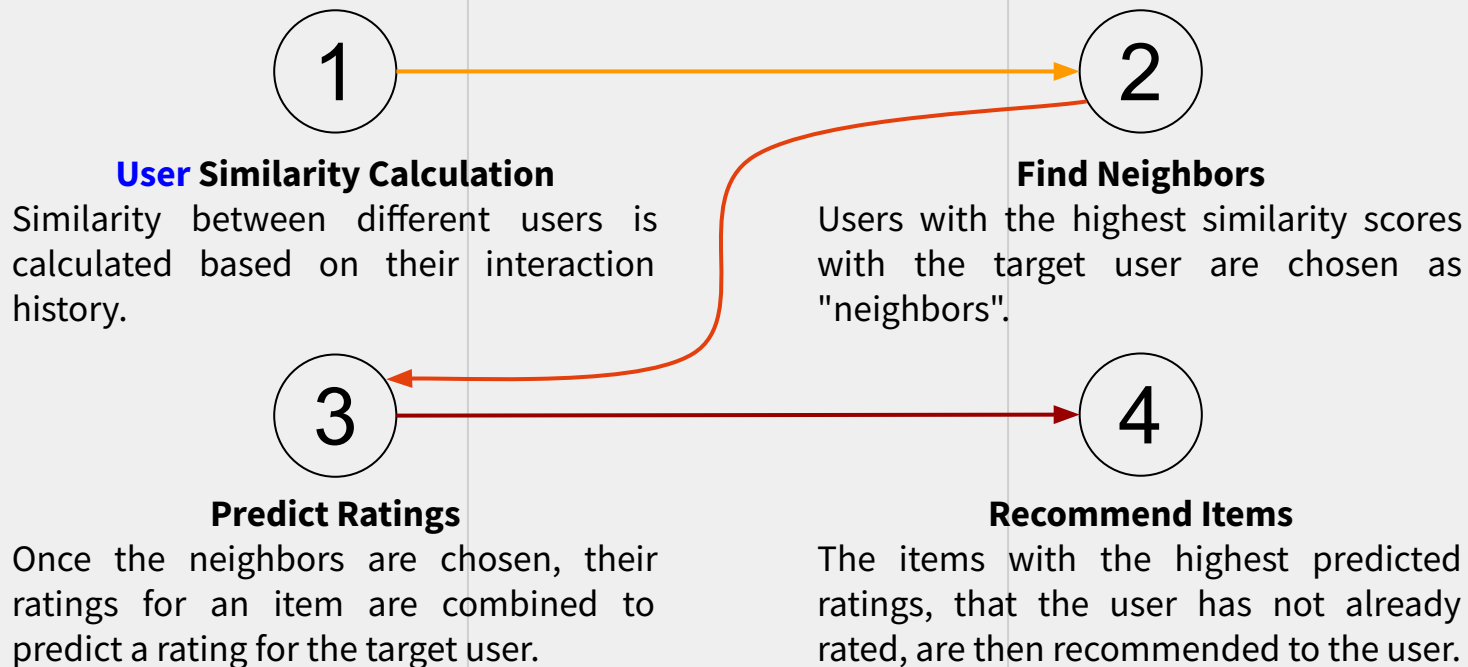
## Model-Based

A type of collaborative filtering that involves building machine learning models based on user-item interaction data to predict a user's interest in different items.

These models can handle the scalability and sparsity issues that are present in memory-based methods but can be more complex and computationally intensive.

# User-Based Memory

**User-Based Memory-Based Collaborative Filtering** operates based on the idea that if two users agree on one issue, they are likely to agree again in the future. The method follows 4 steps:



# User-Based Memory

**User-Based Collaborative Filtering** can suffer from several issues, such as data sparsity and scalability. These problems occur when there are many items but only a few user interactions, making it hard to find similar users, or when the user base is so large that calculating and storing all pairwise similarities between users is computationally expensive.

?

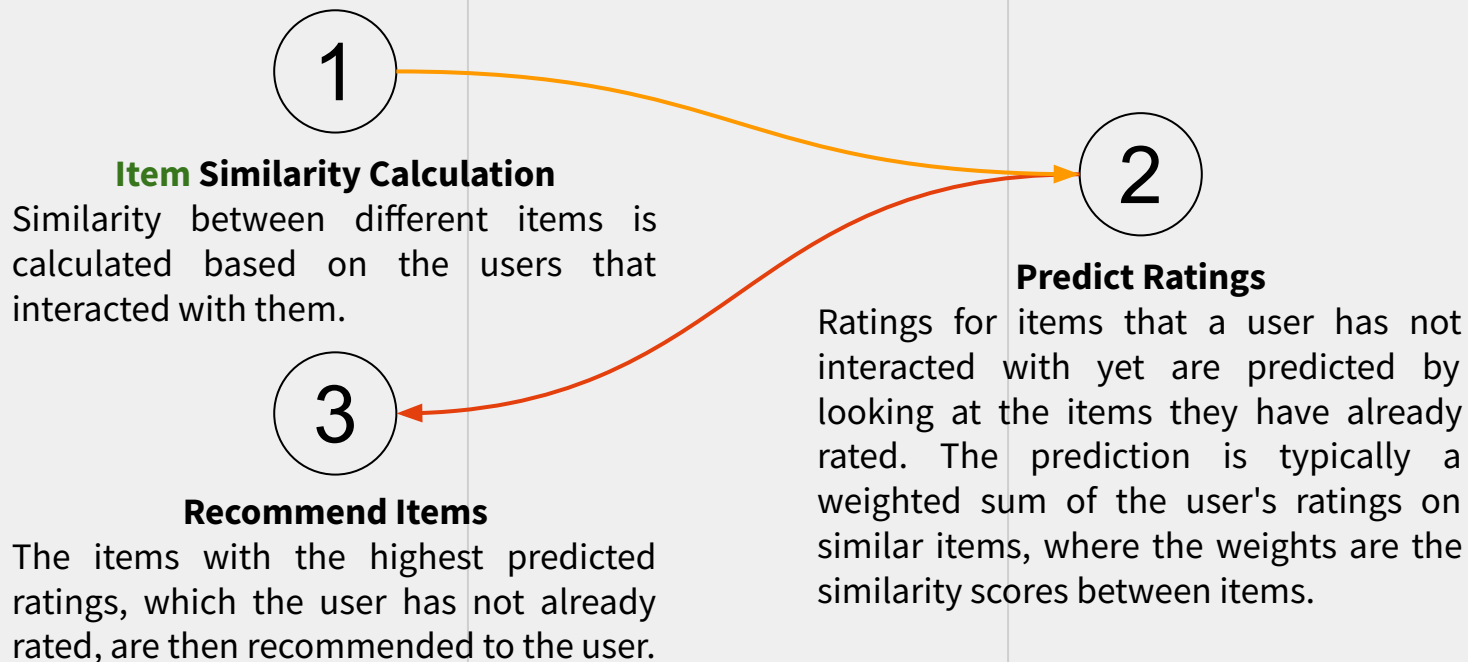
*How many **users** is “too much” for the **user-based** method?*

*There is no universal threshold as it depends on the hardware capacity (computational power and storage), the user activity (if users rate items infrequently, then the user-item matrix will be sparse, which can reduce the computation and storage requirements) and the algorithm/data structure used.*

*As a general rule, it starts to become inefficient with tens of thousands of users and completely intractable beyond millions of users.*

# Item-Based Memory

**Item-Based Memory-Based Collaborative Filtering** focuses on the relationships between items, rather than between users. The idea is that if a user liked a certain item, they are likely to also like items that are similar to it. The method follows 3 steps:

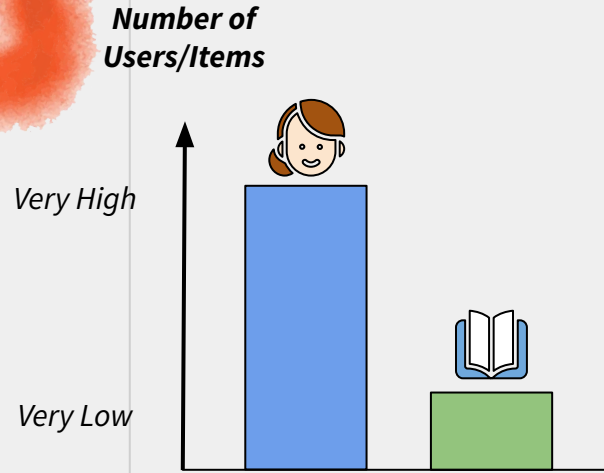


# Item-Based Memory

**Item-Based Collaborative Filtering** typically offers better scalability and performance compared to User-Based Collaborative Filtering, especially when dealing with large numbers of users and items. This is because the relationships between items are more static and don't change as frequently as user-item relationships, so the item-item matrix doesn't need to be updated as often.

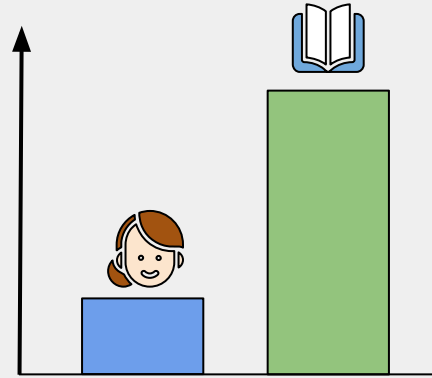
This approach also tends to give more accurate recommendations because it considers the items themselves, not the preferences of similar users, which can sometimes be quite diverse. Furthermore, it can handle sparse data better since it doesn't need to find users who have rated the same items.

# User vs Item



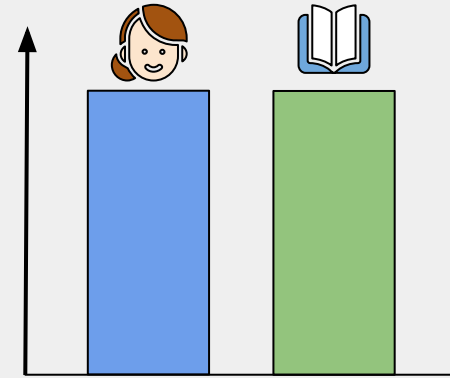
**Item**-Based > **User**-Based

*If the number of users is very high or the user database changes very often.*



**User**-Based > **Item**-Based

*If the number of items is very high or the item database changes very often.*



**Model**-based  
(Or **Item**-Based)

*Items are generally more stable than users.*

# Model-based

**Model-Based collaborative Filtering** builds a mathematical model that captures the relationships between users and items based on their historical interactions. Rather than relying solely on the explicit similarities or correlations between users or items, model-based collaborative filtering aims to learn latent factors or representations that capture the underlying patterns and preferences.

## Matrix Factorization

It decomposes the user-item interaction matrix into lower-dimensional latent factors. Some of the most common methods include *Singular Value Decomposition* and *Alternating Least Squares*.

## Neural Networks

It uses layers of artificial neurons to learn the underlying representations of users and items. Some of the most common methods include *Multi Layer Perceptrons* and *Convolutional Neural Networks*.

## Bayesian

It provides a probabilistic framework for learning and making predictions. Some of the most common methods include *Bayesian Personalized Ranking* or *Bayesian Matrix Factorization*.

Other techniques exist such as Support Vector Machines, Factorization Machines, Hybrid techniques, and many more.

# Matrix Factorization

**Matrix Factorization** decomposes the user-item interaction matrix into lower-dimensional matrices, representing users and items, which are used to make recommendations. It typically involves 4 steps:

## User-Item Matrix Creation

Creation of the User-Item matrix from the interactions (ratings, reviews, history ...)

1

## Latent Factor Learning

An objective function is used to quantify the quality of the approximation made by the factorized matrices. The goal is to find the values that minimize the errors between the true user-item matrix and the approximated one.

3

## Matrix Decomposition

Decomposition of the user-item matrix into lower-dimensional matrices which dimensions are determined by the desired trade-off between model complexity and generalization ability.

2

## Prediction & Recommendation

The latent factors can then be used to predict the missing or unobserved entries in the user-item matrix.

4



# Alternative Least Square

**Alternating Least Squares** (or ALS) is an optimization algorithm commonly used in recommendation systems for collaborative filtering that is particularly effective in dealing with large, sparse user-item matrices. ALS is a matrix factorization model-based technique and thus, it aims to factorize the user-item matrix into lower-dimensional user and item matrices.

The main specificity of the ALS algorithm compared to other factorization techniques is its alternating optimization approach. It alternately fixes one matrix - either the user matrix or the item matrix - and optimizes the other while solving least squares problems.

	Item 1	Item 2	Item 3	Item 4
User 1	NaN	2	NaN	5
User 2	3	NaN	NaN	NaN
User 3	5	NaN	4	NaN
User 4	NaN	NaN	4	3

*User-Item matrix*



	Latent Variables	
User 1	1.5	2
User 2	0.3	1.7
User 3	1.1	1.3
User 4	0.5	1.4

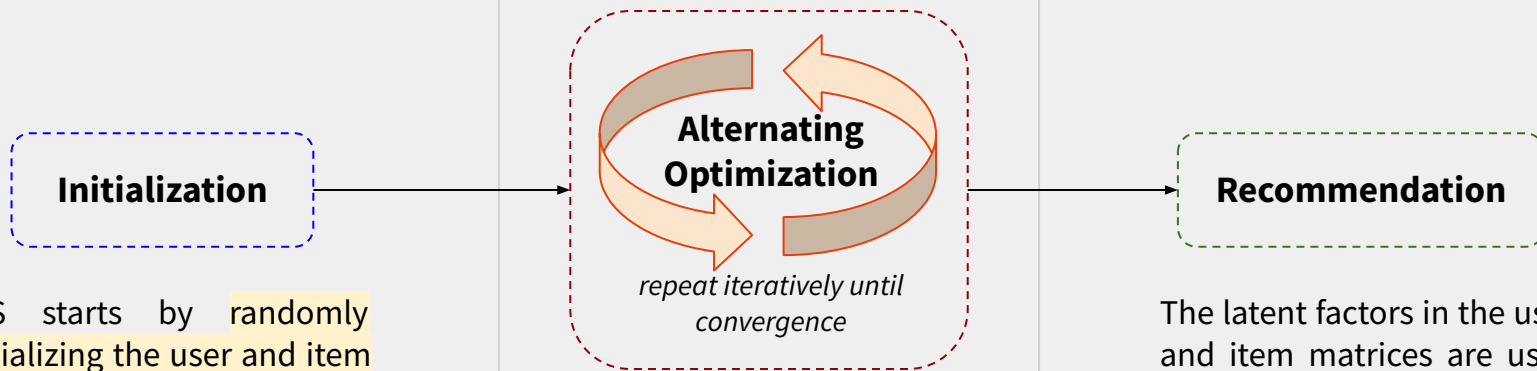
*Lower-Dimensional  
User Matrix*



	Item 1	Item 2	Item 3	Item 4
Latent Variables	1.2	0.8	0.7	1.6
	1.2	0.5	1.5	1.3

*Lower-Dimensional  
Item Matrix*

# Alternative Least Square



ALS starts by randomly initializing the user and item matrices.

These matrices have a lower rank or fewer dimensions compared to the original user-item matrix.

**Step 1** - ALS fixes the user matrix and optimizing the item matrix. It uses the fixed user matrix and the observed user-item interactions to estimate the latent factors for each item using a least square method to minimize the error.

**Step 2** - Then, it fixes the item matrix and proceeds to optimize the user matrix.

Repeat steps 1 and 2 until convergence.

The latent factors in the user and item matrices are used to predict the missing or unobserved interactions in the user-item matrix.

Recommendations are made by selecting the top-ranked items based on these predicted values for a given user.

# Cold Start Problem

The **cold-start problem** refers to a problem that arises when a recommender system encounters new users or items for which it has limited or no historical data or information.

However, the system relies on historical user-item interactions to learn and make accurate predictions. Thus, without sufficient data, it may struggle to generate relevant and personalized recommendations for new users or items.



	Interactions
User 1	52
User 2	23
User 3	42
User 4	15
User 5	1
User 6	2

Cold-Start Users  
*i.e. users with very  
little (or no)  
interactions*

# Cold Start Problem

The **cold-start problem** can be mitigated using hybrid approaches that deal with cold start users and/or items separately.

Content-based, Knowledge-based and popularity based recommender systems can all be used to generate recommendation for cold-start users while the others are generated thanks to a collaborative filtering system.

