Recommendation System

Collaborative Filtering

Digital Vidya

Why Recommender?

"We are leaving the age of information and entering the age of recommendation."

—— Chris Anderson in "The Long Tail"

The Age of Recommendation

Search:

User → Items

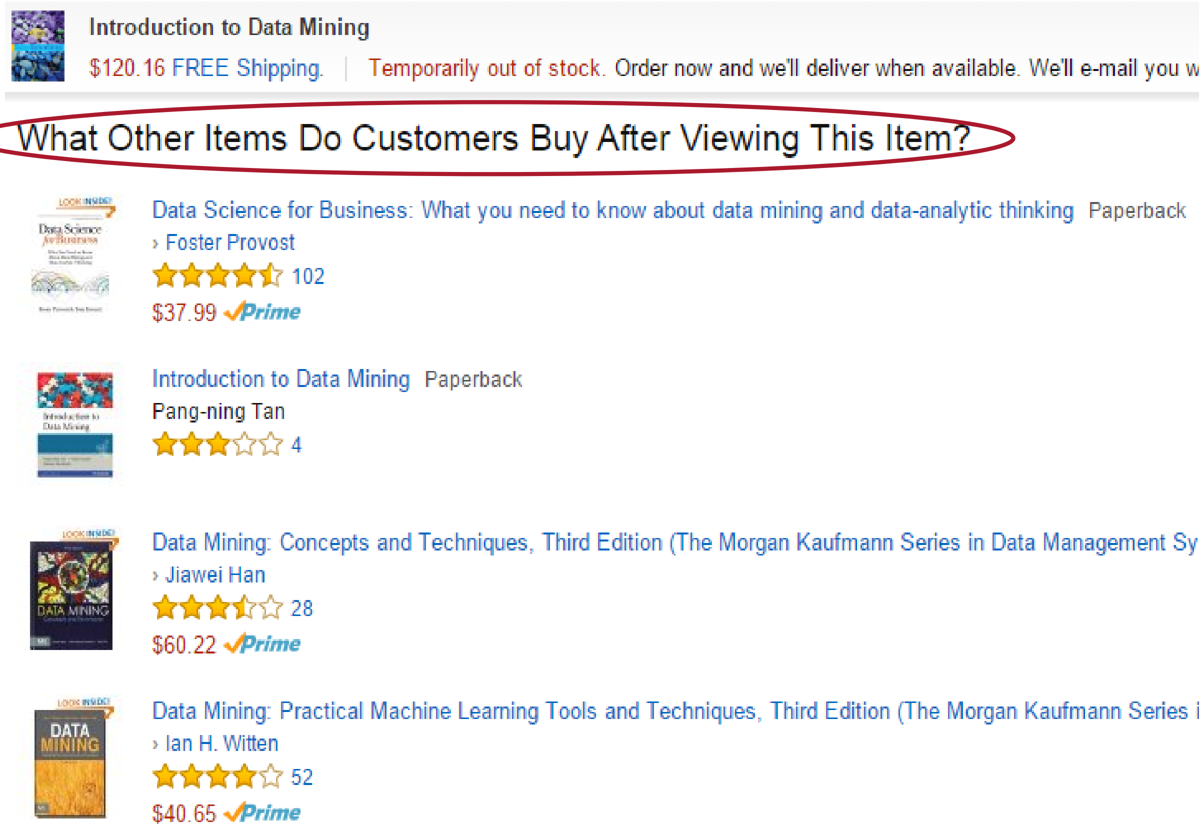
Recommend:

Items → User

Amazon: A personalized online store

Picture from: amazon.com

Amazon: A personalized online store



Picture from: amazon.com

Recommender System

- **Recommendation systems (RS) help to match users with items**
  - Ease information overload
  - Sales assistance (guidance, advisory, persuasion,…)

> *RS are software agents that elicit the interests and preferences of individual consumers [...] and make recommendations accordingly.*
> *They have the potential to support and improve the quality of the decisions consumers make while searching for and selecting products online.*
>
> » (Xiao & Benbasat 2007[1])

- **Different system designs / paradigms**
  - Based on availability of exploitable data
  - Implicit and explicit user feedback
  - Domain characteristics

Recommender Problem

A good recommender

- Show programming titles to a software engineer and baby toys to a new mother.
- Don’t recommend items user already knows or would find anyway.
- Expand user’s taste without offending or annoying him/her…

Challenges

- Huge amounts of data, tens of millions of customers and millions of distinct catalog items.
- Results are required to be returned in real time.
- New customers have limited information.
- Old customers can have a glut of information.
- Customer data is volatile.

Amazon's solution

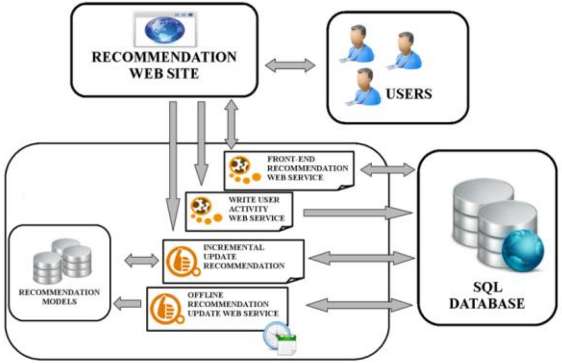
1. Amazon Recommendation Engine

- Amazon's model that implements recommendation algorithm.
- Recommendation algorithm is designed to personalize the online store for each customer.

2. Algorithm feature

- Most recommendation algorithms start by finding a set of similar customers whose purchased and rated items overlap the user's purchased and rated items.
- The Amazon's item-to-item collaborative filtering is focusing on finding similar items instead of similar customers.

3. Recommendation Engine Workflow



Traditional Recommendation Algorithms

Two mostly used traditional algorithms:

1. User Based Collaborative Filtering

2. Cluster Models

User Based Collaborative Filtering

Approach

- Represents a customer as an N-dimensional vector of items
- Vector is positive for purchased or positively rated items and negative for negatively rated items
- Based on cosine similarity: finds similar customers/users

$$similarity(\vec{A}, \vec{B}) = \cos(\vec{A}, \vec{B}) = \frac{\vec{A} \bullet \vec{B}}{\|\vec{A}\| * \|\vec{B}\|}$$

- Generates recommendations based on a few customers who are most similar to the user
- Rank each item according to how many similar customers purchased it

Problems

- **computationally expensive**, ***O(MN)*** in the worst case, where
  - M is the number of customers and
  - N is the number of items
- **dimensionality reduction** can increase the performance, BUT, also **reduce the quality of the recommendation**
- For very large data sets, such as **10 million customers and 1 million items**, the algorithm encounters **severe performance and scaling issues**

Cluster Models

Approach

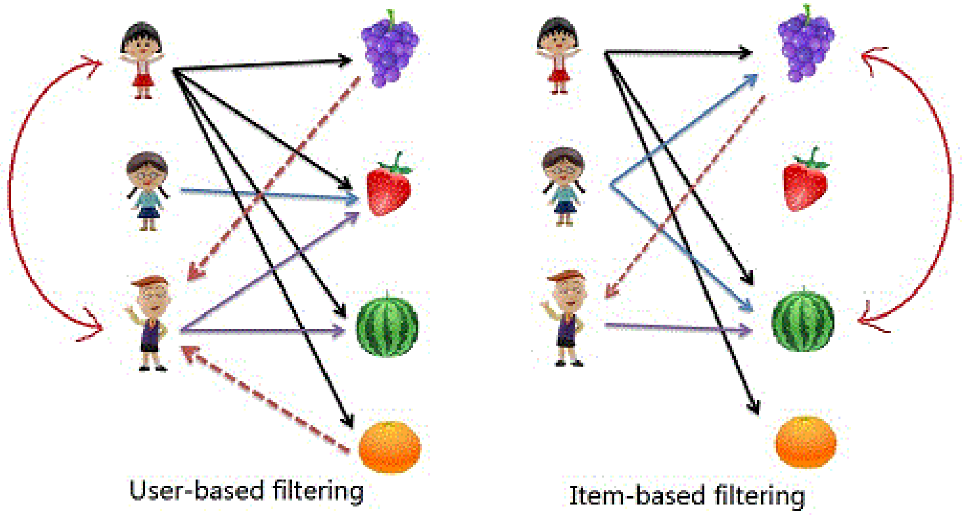
- Divide the customer base into many segments and treat the task as a classification problem
- Assign the user to the segment containing the most similar customers
- Uses the purchases and ratings of the customers in the segment to generate recommendations
- Cluster models have **better online scalability and performance** than collaborative filtering because they compare the user to a controlled number of segments rather than the entire customer base.

Problems

- **Quality of the recommendation is low**
- The recommendations are less relevant because the similar customers that the cluster models find are not the most similar customers
- **To improve quality**, it **needs online segmentation**, which is almost as **expensive as** finding similar customers using **collaborative filtering**

Amazon’s Item-to-Item CF

- **Difference with User-to-User CF**



Amazon’s Item-to-Item CF

Similarity of item i with item 17

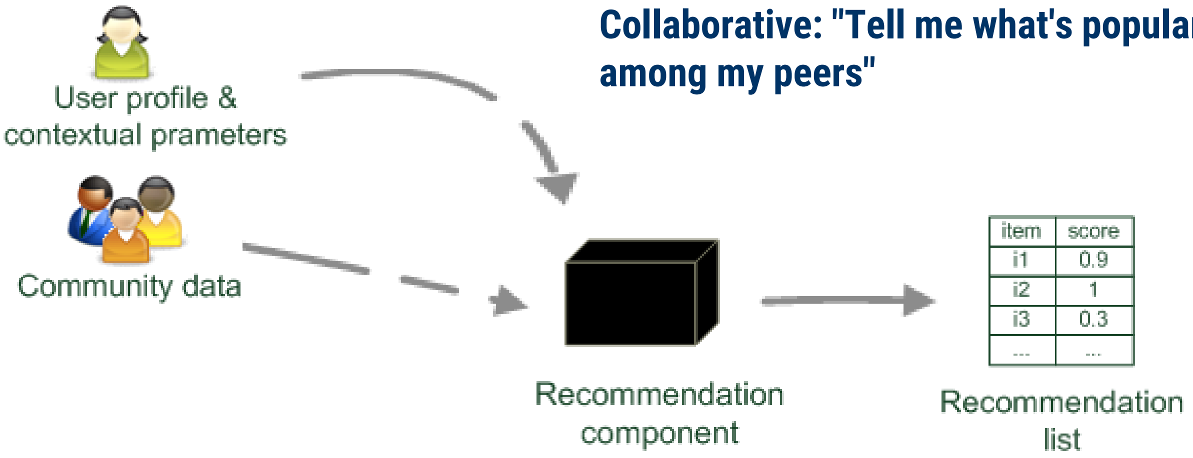
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
,1	,3	,6	,1	,3	,4	,3	,3	,2	,6	,2	,5	,4	,5	,5	,3	1	,3	,5	,4	,2	,4	,4	,5

Items

Users

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
a			1		4	5			4		3					2			4		2				
b			4							3							5	1		3					
c		5		4			4						3		5					4		5			
d								3				5				3			4		2			3	
e		3					5			4	5				5					1			5	4	
f			4				1		3	5		4	1		5	4	4		4				3		
g	2	4				4		2			5		1	4	5		4	2	4		5			4	
h			2		1		4		3	5		4	2		5	4	5						5		
i		1					3			5				5		4	4		5			4		3	
j			4			4				5			1		5		4		4				4		
k		5				4			2		5		1	5		4		2		4				2	
l					3			3				4	1		4		4	2	4					3	
m	5		3					5	3		5	4		5	5	3			4	4	5	4		4	
n			1		4	5				4	5		1	5		4		3		4		4	3		
o			4			4				5		4		5			4	2		5		5		3	
p				4			5								5	4		2	4	4	5	4		2	
q					3			3					1	5		4	4		4			4		3	
r		4			1	4		2					2		5		4				5	4		4	
s			2		4		4			5			1			4		2	4		4		5		
t		1		4			3					4		5	5		4			4				3	
u			2		1		4		3				1		5	4		2	4		5	4			
v					4	5				4	3		5			2					2			5	
w				2			2		3			5			4	5		4	2		3	4			
x	4			5				3		3				4	5					1					
y			1			3				2	3						3	3		5		4			

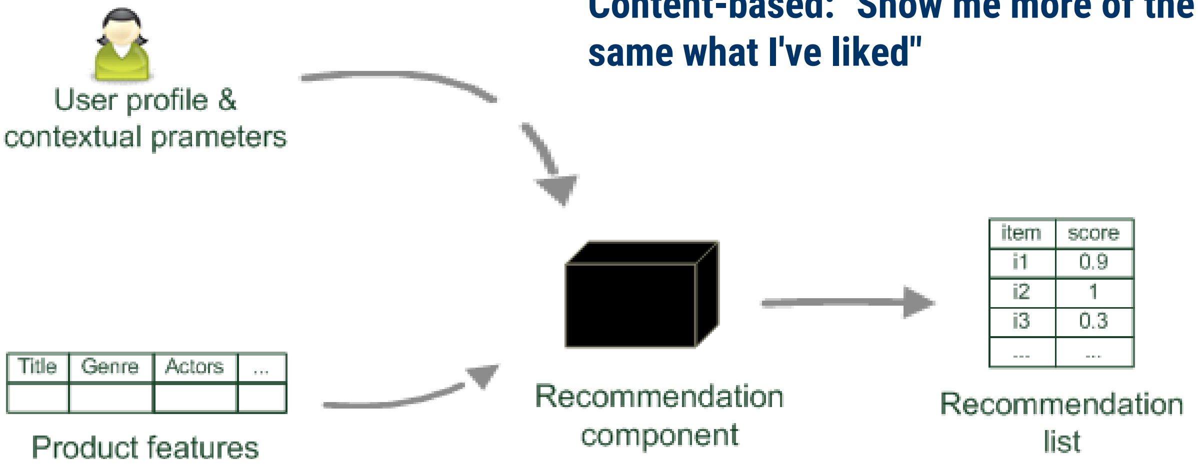
Collaborative Filtering based recommender systems



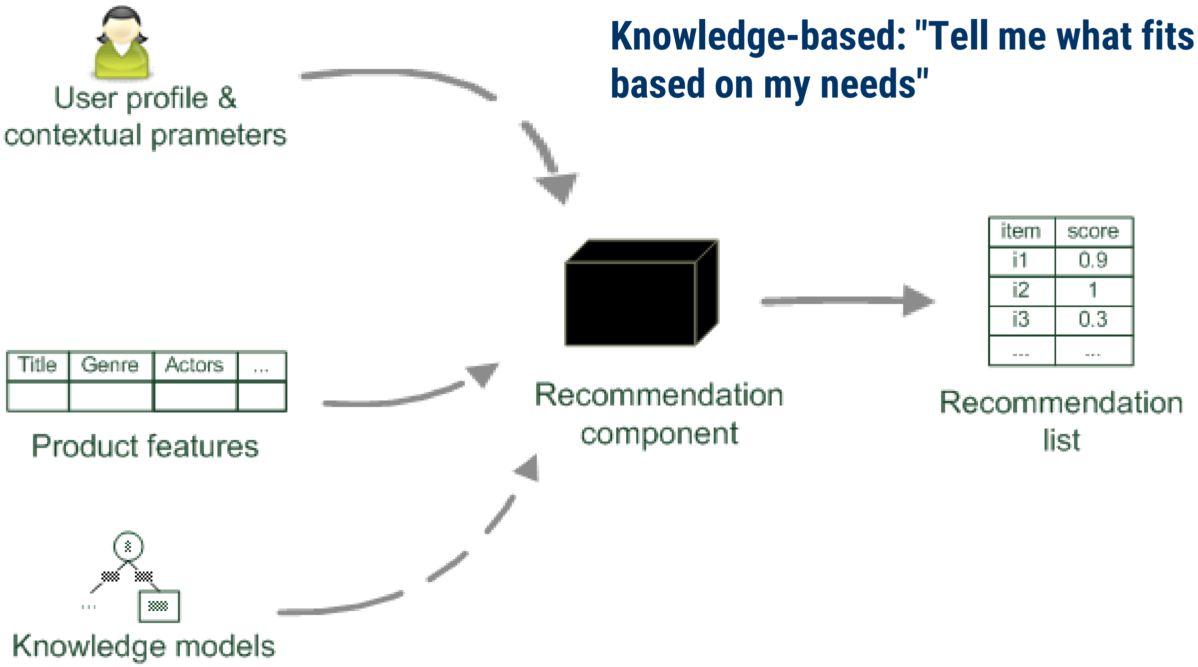
Collaborative: "Tell me what's popular among my peers"

Content based recommender systems

Content-based: "Show me more of the same what I've liked"



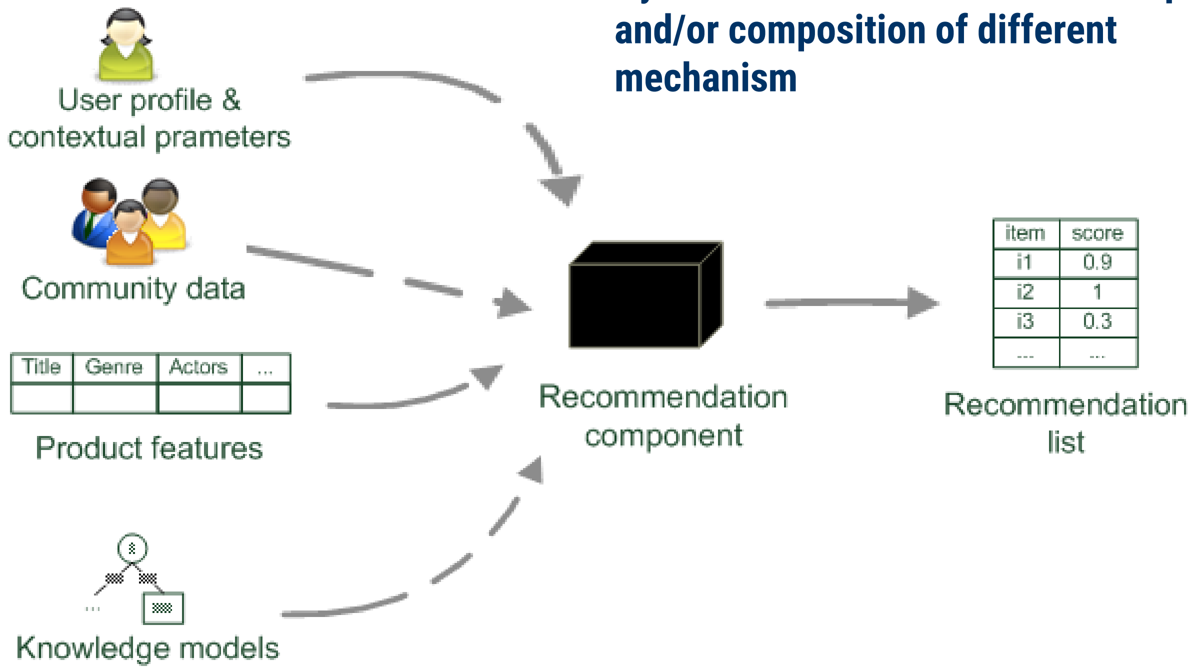
Knowledge based recommender systems



Knowledge-based: "Tell me what fits based on my needs"

Hybrid recommender systems

Hybrid: combinations of various inputs and/or composition of different mechanism



Collaborative Filtering (CF)

- **The most prominent approach to generate recommendations**
  - used by large, commercial e-commerce sites
  - well-understood, various algorithms and variations exist
  - applicable in many domains (book, movies, DVDs, ..)
- **Approach**
  - use the "wisdom of the crowd" to recommend items

- **Basic assumption and idea**
  - Users give ratings to catalog items (implicitly or explicitly)
  - Customers who had similar tastes in the past, will have similar tastes in the future

Pure CF Approaches

- **Input**
  - Only a matrix of given user–item ratings
- **Output types**
  - A (numerical) prediction indicating to what degree the current user will like or dislike a certain item
  - A top-N list of recommended items

User-based nearest-neighbor collaborative filtering (1)

- **The basic technique**
  - Given an "active user" (Alice) and an item i not yet seen by Alice
    - find a set of users (peers/nearest neighbors) who liked the same items as Alice in the past **and** who have rated item i
    - use, e.g. the average of their ratings to predict, if Alice will like item i
    - do this for all items Alice has not seen and recommend the best-rated
- **Basic assumption and idea**
  - If users had similar tastes in the past they will have similar tastes in the future
  - User preferences remain stable and consistent over time

User-based nearest-neighbor collaborative filtering (2)

- **Example**
  - A database of ratings of the current user, Alice, and some other users is given:

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

  - Determine whether Alice will like or dislike *Item5*, which Alice has not yet rated or seen

User-based nearest-neighbor collaborative filtering (3)

- **Some first questions**
  - How do we measure similarity?
  - How many neighbors should we consider?
  - How do we generate a prediction from the neighbors' ratings?

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Measuring user similarity (1)

- **A popular similarity measure in user-based CF: Pearson correlation**

  a, b : users

  $r_{a,p}$: rating of user a for item p

  P : set of items, rated both by a and b

  - Possible similarity values between -1 and 1

$$\boldsymbol{sim(a,b)} = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

Measuring user similarity (2)

- **A popular similarity measure in user-based CF: Pearson correlation**

  a, b : users

  $r_{a,p}$: rating of user a for item p

  P : set of items, rated both by a and b

  – Possible similarity values between -1 and 1

	Item1	Item2	Item3	Item4	Item5	
Alice	5	3	4	4	?	
User1	3	1	2	3	3	sim = 0,85
User2	4	3	4	3	5	sim = 0,00
User3	3	3	1	5	4	sim = 0,70
User4	1	5	5	2	1	sim = -0,79

Making predictions

- A common prediction function:

$$\boldsymbol{pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}}$$

- Calculate, whether the neighbors' ratings for the unseen item i are higher or lower than their average
- Combine the rating differences – use the similarity with a as a weight
- Add/subtract the neighbors' bias from the active user's average and use this as a prediction

Item-based collaborative filtering

- **Basic idea:**
  - Use the similarity between items (and not users) to make predictions
- **Example:**
  - Look for items that are similar to Item5
  - Take Alice's ratings for these items to predict the rating for Item5

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

The cosine similarity measure

- **Produces better results in item-to-item filtering**
- **Ratings are seen as vector in n-dimensional space**
- **Similarity is calculated based on the angle between the vectors**

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

- **Adjusted cosine similarity**
  - take average user ratings into account, transform the original ratings
  - U: set of users who have rated both items a and b

$$sim(\vec{a}, \vec{b}) = \frac{\sum_{u \in U}(r_{u,a} - \overline{r_u})(r_{u,b} - \overline{r_u})}{\sqrt{\sum_{u \in U}(r_{u,a} - \overline{r_u})^2} \sqrt{\sum_{u \in U}(r_{u,b} - \overline{r_u})^2}}$$

Making predictions

- A common prediction function:

$$pred(u,p) = \frac{\sum_{i \in ratedItem(u)} sim(i,p) * r_{u,i}}{\sum_{i \in ratedItem(u)} sim(i,p)}$$

- Neighborhood size is typically also limited to a specific size
- Not all neighbors are taken into account for the prediction
- An analysis of the MovieLens dataset indicates that "in most real-world situations, a neighborhood of 20 to 50 neighbors seems reasonable" (Herlocker et al. 2002)

Explicit ratings

- Probably the most precise ratings
- Most commonly used (1 to 5, 1 to 7 Likert response scales)
- Research topics
  - Optimal granularity of scale; indication that 10-point scale is better accepted in movie dom.
  - An even more fine-grained scale was chosen in the joke recommender discussed by Goldberg et al. (2001), where a continuous scale (from −10 to +10) and a graphical input bar were used
    - No precision loss from the discretization
    - User preferences can be captured at a finer granularity
    - Users actually "like" the graphical interaction method
  - Multidimensional ratings (multiple ratings per movie such as ratings for actors and sound)
- Main problems
  - Users not always willing to rate many items
    - number of available ratings could be too small → sparse rating matrices → poor recommendation quality
  - How to stimulate users to rate more items?

Implicit ratings

- Typically collected by the web shop or application in which the recommender system is embedded
- When a customer buys an item, for instance, many recommender systems interpret this behavior as a positive rating
- Clicks, page views, time spent on some page, demo downloads …
- Implicit ratings can be collected constantly and do not require additional efforts from the side of the user
- Main problem
  - One cannot be sure whether the user behavior is correctly interpreted
  - For example, a user might not like all the books he or she has bought; the user also might have bought a book for someone else
- Implicit ratings can be used in addition to explicit ones; question of correctness of interpretation

Data sparsity problems

- **Cold start problem**
  - How to recommend new items? What to recommend to new users?
- **Straightforward approaches**
  - Ask/force users to rate a set of items
  - Use another method (e.g., content-based, demographic or simply non-personalized) in the initial phase
  - Default voting: assign default values to items that only one of the two users to be compared has rated (Breese et al. 1998)
- **Alternatives**
  - Use better algorithms (beyond nearest-neighbor approaches)
  - Example:
    - In nearest-neighbor approaches, the set of sufficiently similar neighbors might be too small to make good predictions
    - Assume "transitivity" of neighborhoods

Thank You