COMP9313: Big Data Management

Lecturer: Xin Cao

Course web site: http://www.cse.unsw.edu.au/~cs9313/

Chapter 11: Recommender Systems

Recommendations

Search

Recommendations

Items

Products, web sites, blogs, news items, …

Examples:

amazon.com

PANDORA

StumbleUpon

del.icio.us

NETFLIX

movielens
helping you find the *right* movies

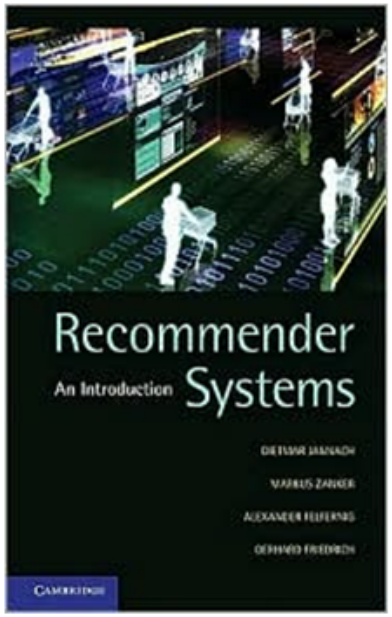
last·fm
the social music revolution

Google News

YouTube

XBOX LIVE

Recommender Systems



Recommender Systems: An Introduction

by Dietmar Jannach, Markus Zanker, Alexander Felfernig, Gerhard Friedrich

AVERAGE CUSTOMER RATING:

(Be the first to review)

Gefällt mir Registrieren, um sehen zu können, was deinen Freunden gefällt.

FORMAT:
Hardcover

NOOKbook (eBook) - not available
Tell the publisher you want this in NOOKbook format

NEW FROM BN.COM

~~$65.00~~ List Price
$52.00 Online Price
(You Save 20%)

Add to Cart

NEW & USED FROM OUR

New starting at **$56.46** (You S
Used starting at **$51.98** (You S

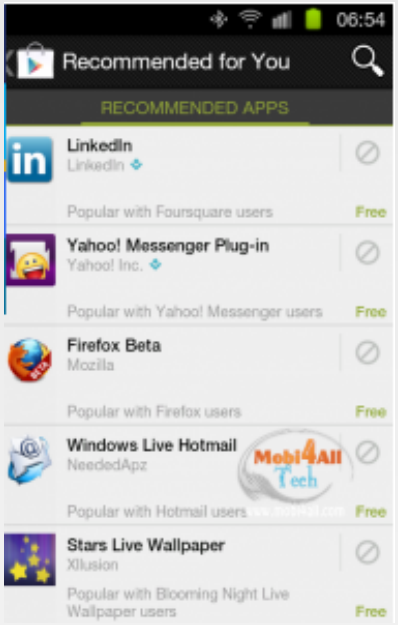
See All Prices

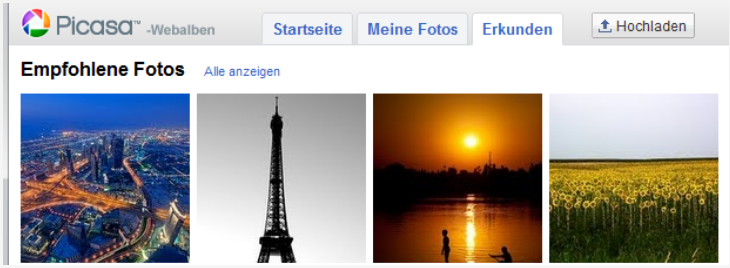
Table of Contents

Customers who bought this also bought

Recommender Systems

- Application areas





Why using Recommender Systems?

- Value for the customer
  - Find things that are interesting
  - Narrow down the set of choices
  - Help me explore the space of options
  - Discover new things
  - Entertainment
  - …
- Value for the provider
  - Additional and probably unique personalized service for the customer
  - Increase trust and customer loyalty
  - Increase sales, click trough rates, conversion etc.
  - Opportunities for promotion, persuasion
  - Obtain more knowledge about customers
  - …

Real-world Check

- Myths from industry
  - Amazon.com generates X percent of their sales through the recommendation lists ($30 < X < 70$)
  - Netflix (DVD rental and movie streaming) generates X percent of their sales through the recommendation lists ($30 < X < 70$)

Recommender systems

- RS seen as a function
- Given:
  - User model (e.g. ratings, preferences, demographics, situational context)
  - Items (with or without description of item characteristics)
- Find:
  - Relevance score. Used for ranking.
- Finally:
  - Recommend items that are assumed to be relevant
- But:
  - Remember that relevance might be context-dependent
  - Characteristics of the list itself might be important (diversity)

Formal Model

- X = set of Customers
- S = set of Items
- Utility function u: $X \times S \rightarrow R$
  - R = set of ratings
  - R is a totally ordered set
  - e.g., 0-5 stars, real number in [0,1]
- Utility Matrix

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4

Key Problems

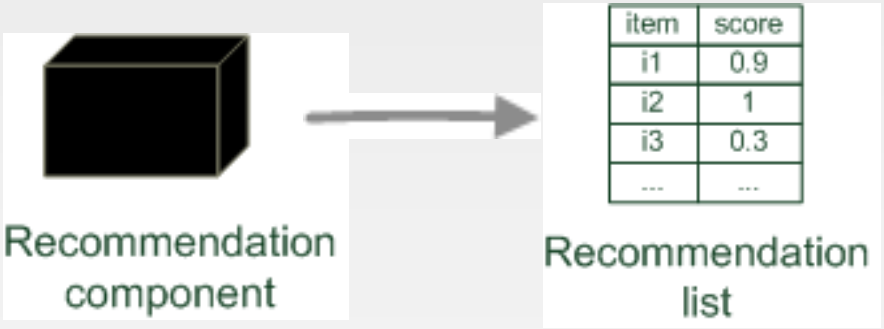
- Gathering “known” ratings for matrix
  - How to collect the data in the utility matrix
- Extrapolate unknown ratings from the known ones
  - Mainly interested in high unknown ratings
    - We are not interested in knowing what you don’t like but what you like
- Evaluating extrapolation methods
  - How to measure success/performance of recommendation methods

Gathering Ratings

- Explicit
  - Ask people to rate items
  - Doesn't work well in practice – people can't be bothered
- Implicit
  - Learn ratings from user actions
    - E.g., purchase implies high rating
  - What about low ratings?

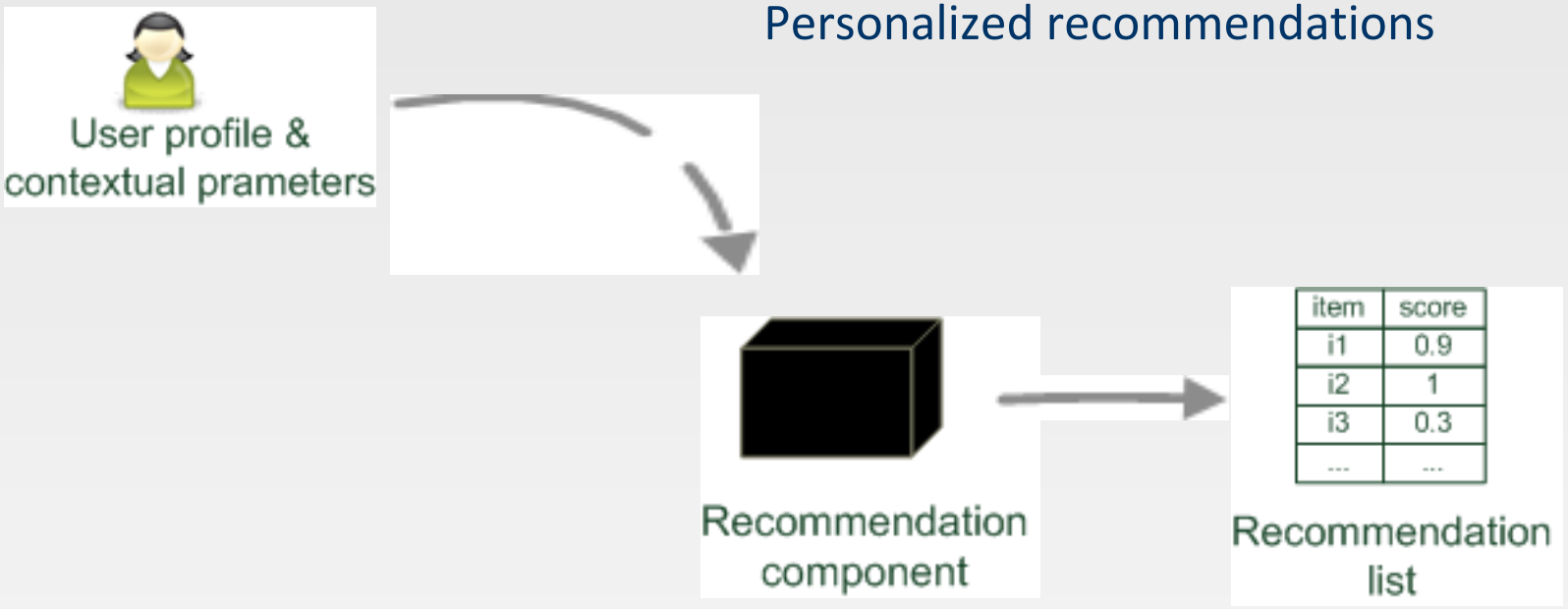
Paradigms of recommender systems

Recommender systems reduce information overload by estimating relevance

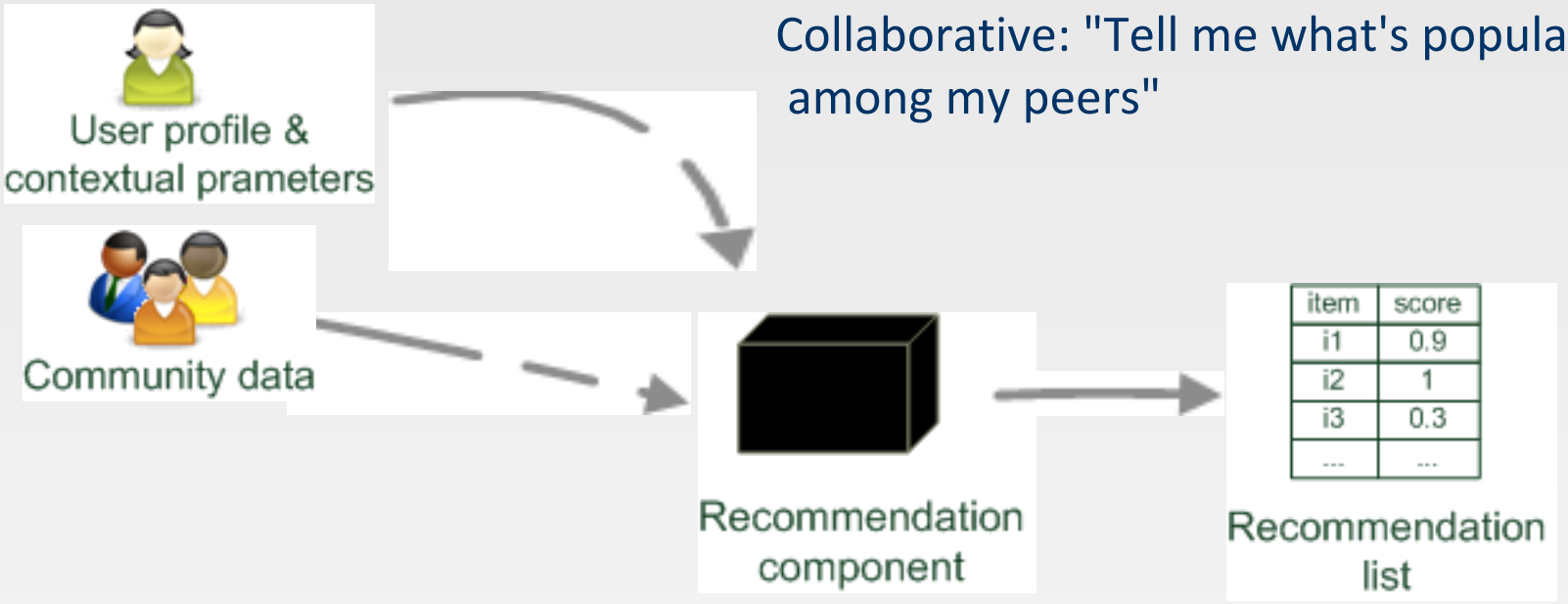


Paradigms of recommender systems

Personalized recommendations



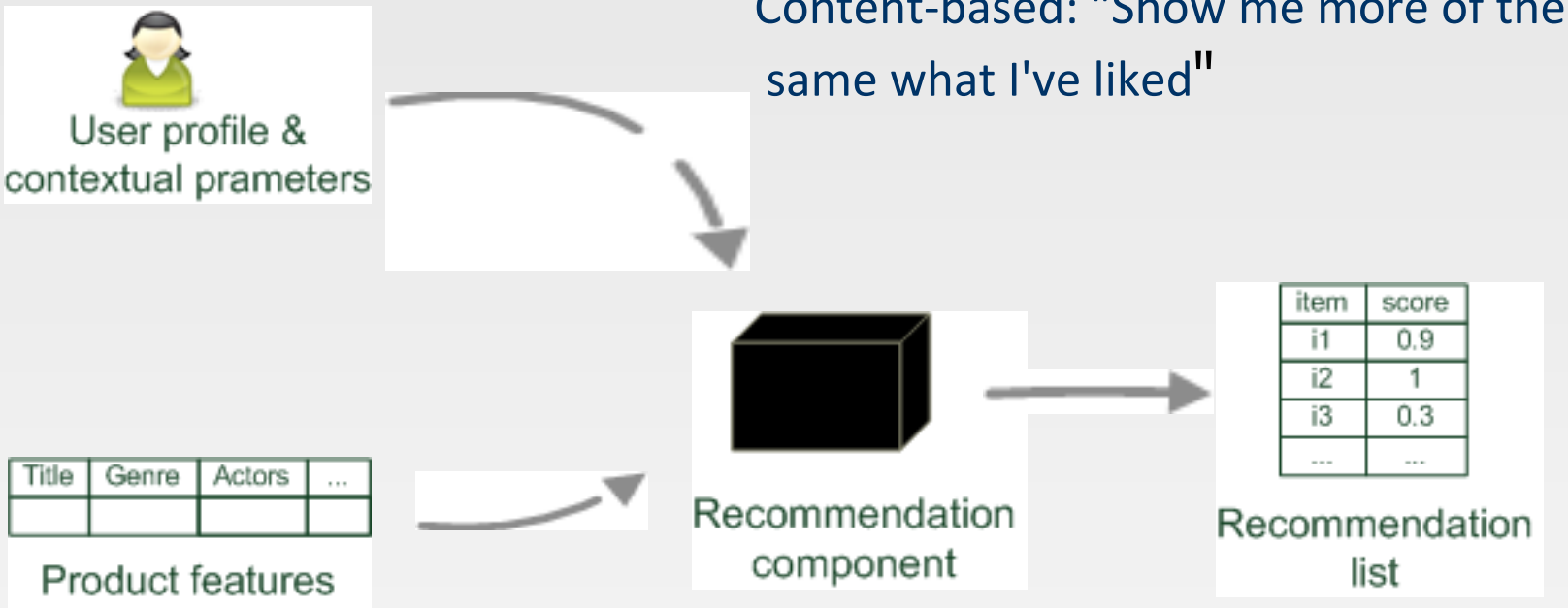
Paradigms of recommender systems



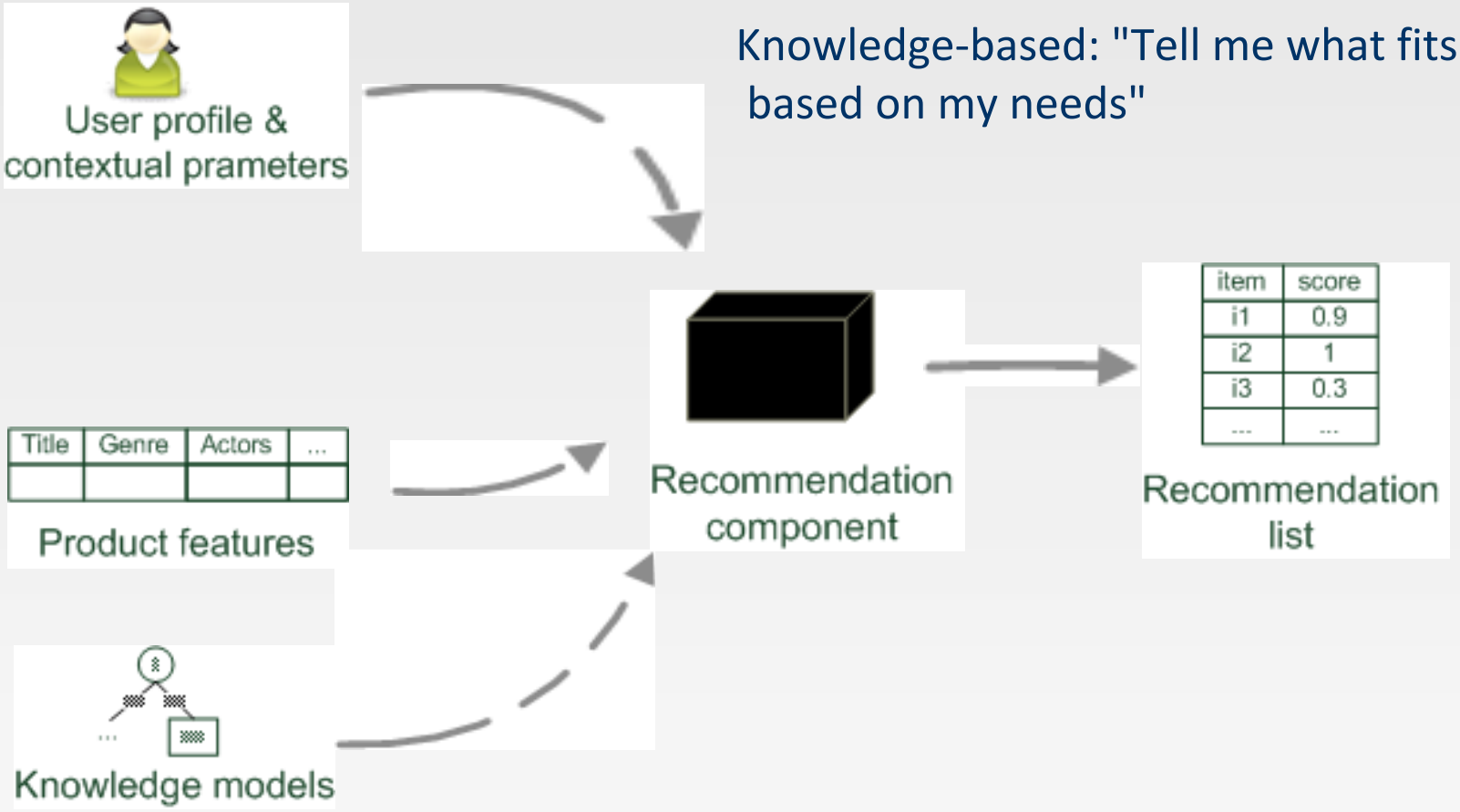
Collaborative: "Tell me what's popular among my peers"

Paradigms of recommender systems

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

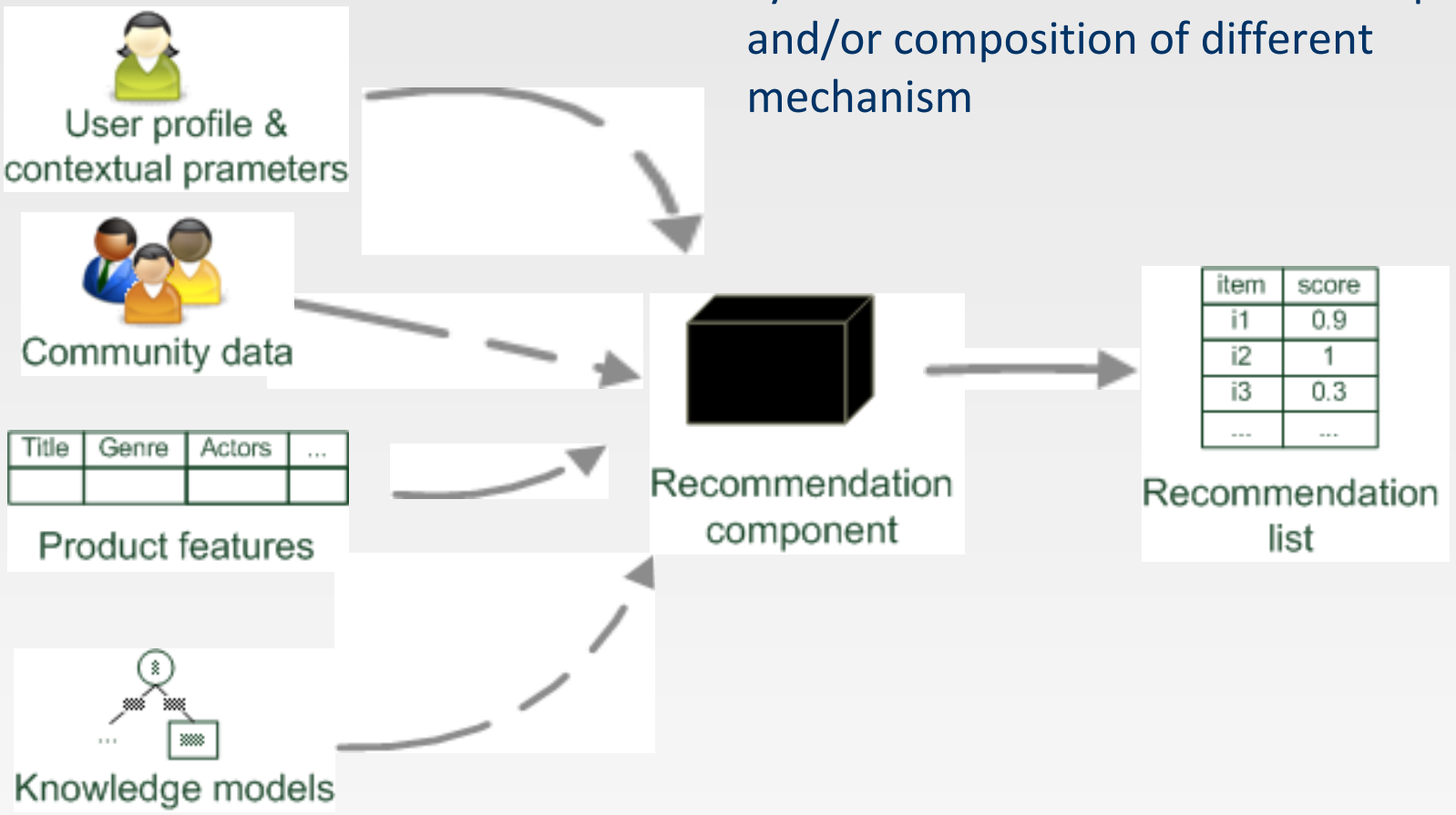


Paradigms of recommender systems



Paradigms of recommender systems

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



Recommender systems: basic techniques

	Pros	Cons
Content-based	No community required, comparison between items possible	Content descriptions necessary, cold start for new users, no surprises
Collaborative	No knowledge-engineering effort, serendipity of results, learns market segments	Requires some form of rating feedback, cold start for new users and new items
Knowledge-based	Deterministic recommendations, assured quality, no cold-start, can resemble sales dialogue	Knowledge engineering effort to bootstrap, basically static, does not react to short-term trends

Content-based Recommendations

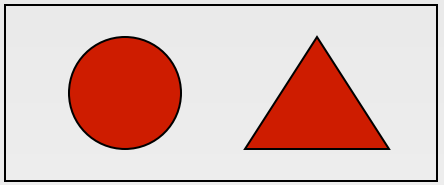
- **Main idea:** Recommend items to customer ***x*** similar to previous items rated highly by ***x***

- What do we need:
  - Some information about the available items such as the genre ("content")
  - Some sort of *user profile* describing what the user likes (the preferences)

- ***Example:***
  - Movie recommendations:
    - Recommend movies with same actor(s), director, genre, …
  - Websites, blogs, news:
    - Recommend other sites with “similar” content

Plan of Action

likes



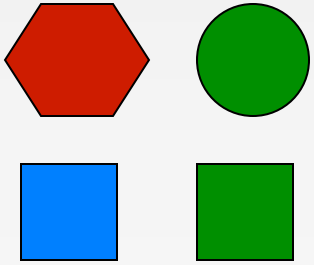
Item profiles

build

Red
Circles
Triangles

User profile

match



recommend

Item Profiles

- For each item, create an **item profile**
- Profile is a set (vector) of features
  - **Movies:** author, title, actor, director,…
  - **Text:** Set of “important” words in document
- How to pick important features?
  - Usual heuristic from text mining is **TF-IDF** (Term frequency * Inverse Doc Frequency)
    - **Term … Feature**
    - **Document … Item**

Term-Frequency - Inverse Document Frequency (TF-IDF)

- Compute the overall importance of keywords
  - Given a keyword i and a document j

    $$TF\text{-}IDF\ (i,j) = TF(i,j) * IDF(i)$$

- Term frequency (TF)
  - Let $freq(i,j)$ number of occurrences of keyword *i* in document *j*
  - Let $maxOthers(i,j)$ denote the highest number of occurrences of another keyword of *j*
  - $TF(i,j) = freq(i,j)/maxOthers(i,j)$
- Inverse Document Frequency (IDF)
  - N: number of all recommendable documents
  - n(i): number of documents in which keyword *i* appears
  - $IDF(i) = logN/n(i)$

User Profiles and Prediction

- User profile possibilities:
  - Weighted average of rated item profiles
  - **Variation:** weight by difference from average rating for item
  - …

- **Prediction heuristic:**
  - Given user profile $\boldsymbol{x}$ and item profile $\boldsymbol{i}$, estimate

$$u(\boldsymbol{x},\boldsymbol{i}) = \cos(\boldsymbol{x},\boldsymbol{i}) = \boldsymbol{x}\cdot\boldsymbol{i}/|\boldsymbol{x}|\cdot|\boldsymbol{i}|$$

Pros: Content-based Approach

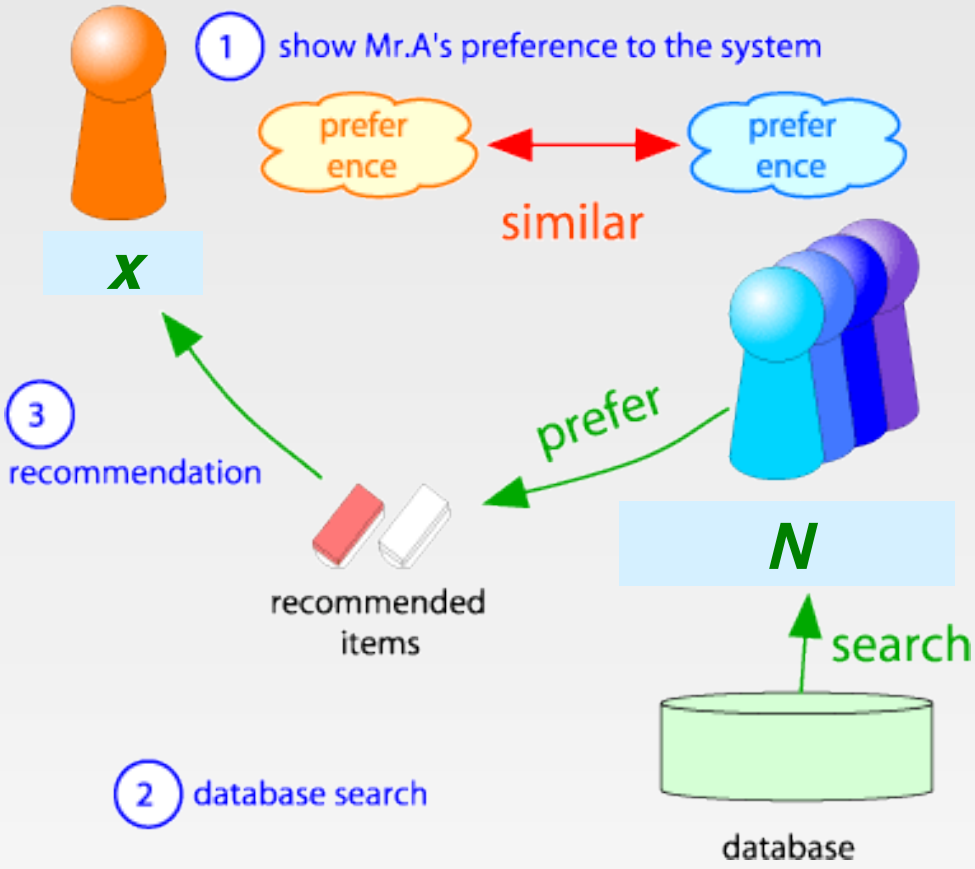
- **+: No need for data on other users**
  - No cold-start or sparsity problems
- **+: Able to recommend to users with unique tastes**
- **+: Able to recommend new & unpopular items**
  - No first-rater problem
- **+: Able to provide explanations**
  - Can provide explanations of recommended items by listing content-features that caused an item to be recommended

Cons: Content-based Approach

- **–: Finding the appropriate features is hard**
  - E.g., images, movies, music
- **–: Recommendations for new users**
  - **How to build a user profile?**
- **–: Overspecialization**
  - Never recommends items outside user's content profile
  - People might have multiple interests
  - **Unable to exploit quality judgments of other users**

Collaborative Filtering

- Consider user ***x***
- Find set ***N*** of other users whose ratings are "**similar**" to ***x***'s ratings
- Estimate ***x***'s ratings based on ratings of users in ***N***



User-based Nearest-Neighbor Collaborative Filtering

- The basic technique:
  - Given an "active user" (Alice) and an item ***i*** not yet seen by Alice
  - The *goal is to estimate Alice's rating for this item*, e.g., by
    - find a set of users (peers) 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

	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

User-based Nearest-Neighbor Collaborative Filtering (Cont')

- 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

Finding "Similar" Users

r_x = [*, _, _, *, ***]
r_y = [*, _, **, **, _]

- Let r_x be the vector of user x's ratings
- **Jaccard similarity measure** $||r_x \cap r_y|| / ||r_x \cup r_y||$
  - **Problem:** Ignores the value of the rating
- **Cosine similarity measure**
  - sim(x, y) = cos(r_x, r_y) = $r_x \cdot r_y / ||r_x|| \cdot ||r_y||$
  - **Problem:** Treats missing ratings as "negative"
- **Pearson correlation coefficient**
  - S_{xy} = items rated by both users x and y

r_x, r_y as sets:
r_x = {1, 4, 5}
r_y = {1, 3, 4}

r_x, r_y as points:
r_x = {1, 0, 0, 1, 3}
r_y = {1, 0, 2, 2, 0}

$$sim(x,y) = \sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)(r_{ys} - \bar{r}_y) / \sqrt{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)^2} \quad \sqrt{\sum_{s \in S_{xy}}}$$

$\bar{r}_x$, $\bar{r}_y$ … avg. rating of **x**, **y**

Similarity Metric

Cosine similarity:

$sim(x, y) = \sum i \uparrow \blacksquare r \downarrow xi \cdot r \downarrow yi \; / v$

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- **Intuitively we want: sim(*A*, *B*) > sim(*A*, *C*)**
- **Jaccard similarity:** 1/5 < 2/4
- **Cosine similarity:** 0.380 > 0.322

$$\frac{4 \times 5}{\sqrt{4^2 + 5^2 + 1^2}\sqrt{5^2 + 5^2 + 4^2}} = 0.380$$

  - Considers missing ratings as “negative”
  - **Solution: subtract the (row) mean**

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	2/3			5/3	−7/3		
B	1/3	1/3	−2/3				
C				−5/3	1/3	4/3	
D		0					0

sim A,B vs. A,C:
0.092 > -0.559

Notice cosine sim. is correlation when data is centered at 0

Similarity Metric (Cont')

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

$$sim(x,y)=\sum_{s\in S_{xy}}(r_{xs}-r_x)(r_{ys}-r_y)\,/\sqrt{\sum_{s\in S_{xy}}(r_{xs}-r_x)^2}\quad\sqrt{}$$

  - Possible similarity values between -1 and 1;

	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

sim = 0,85
sim = 0,70
sim = -0,79

Rating Predictions

From similarity metric to recommendations:

- Let r_x be the vector of user x's ratings
- Let N be the set of k users most similar to x who have rated item i
- **Prediction for item *s* of *user x*:**
  - $r_{xi} = 1/k \sum_{y \in N} r_{yi}$
  - $r_{xi} = \sum_{y \in N} s_{xy} \cdot r_{yi} \; / \sum_{y \in N} s_{xy}$
  - Other options?
- **Many other tricks possible…**

Shorthand:

$s_{xy} = sim(x,y)$

Memory-based and Model-based Approaches

- User-based CF is said to be "memory-based"
  - the rating matrix is directly used to find neighbors / make predictions
  - does not scale for most real-world scenarios
  - large e-commerce sites have tens of millions of customers and millions of items
- Model-based approaches
  - based on an offline pre-processing or "model-learning" phase
  - at run-time, only the learned model is used to make predictions
  - models are updated / re-trained periodically
  - large variety of techniques used
  - model-building and updating can be computationally expensive

Item-Item Collaborative Filtering

- **So far: User-user collaborative filtering**
- **Another view: Item-item**
  - Basic idea:
    - Use the similarity between items (and not users) to make predictions
  - For item ***i***, find other similar items
  - Estimate rating for item ***i*** based on ratings for similar items
  - Can use same similarity metrics and prediction functions as in user-user model

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

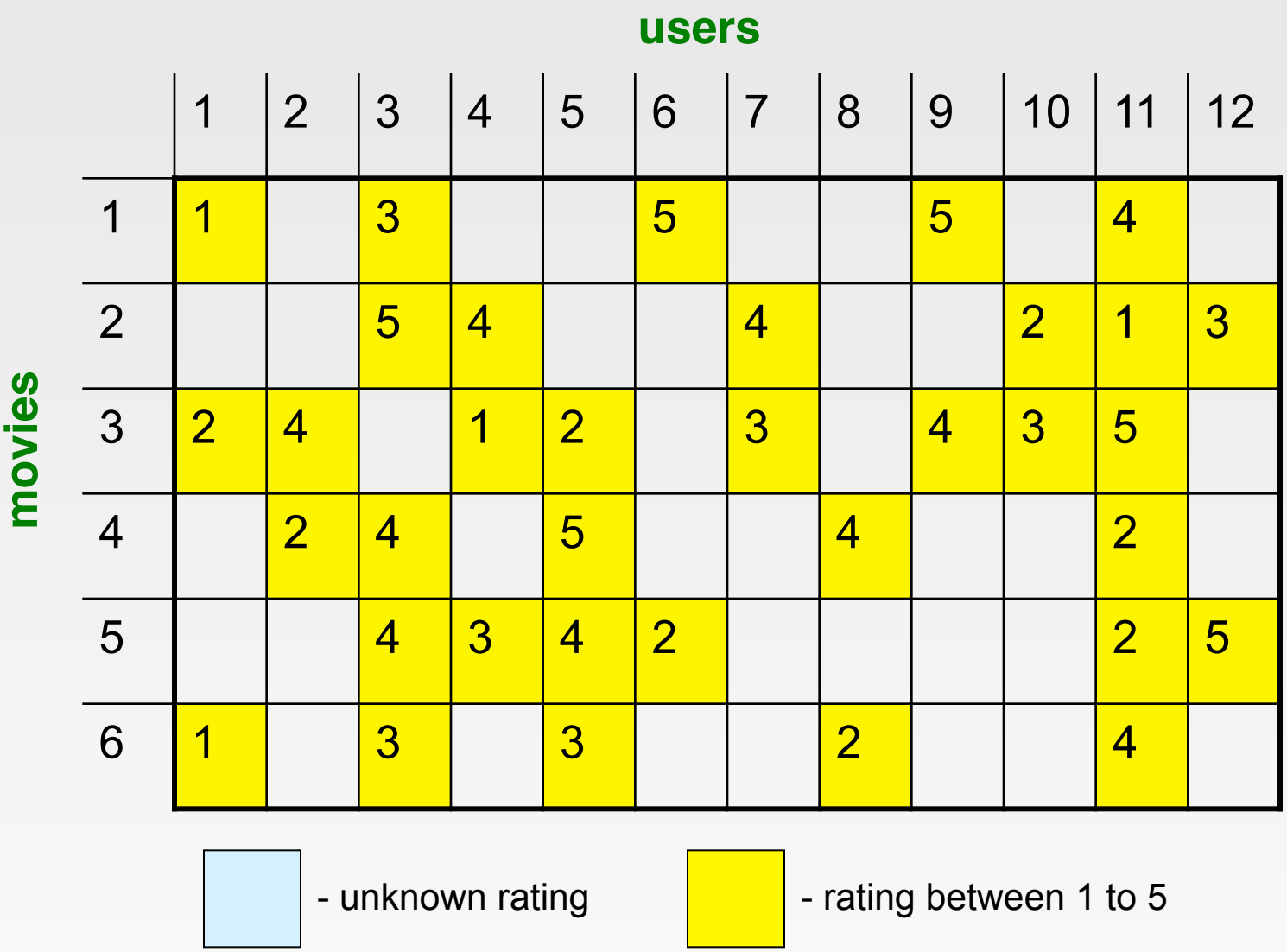
s_{ij}… similarity of items ***i*** and ***j***
r_{xj}…rating of user ***u*** on item ***j***
N(i;x)… set items rated by ***x*** similar to ***i***

Item-Item Collaborative Filtering

- 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

Item-Item CF (INI=2)



Item-Item CF (INI=2)

users

movies

	1	2	3	4	**5**	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- estimate rating of movie **1** by user **5**

Item-Item CF (INI=2)

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		0.41
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		0.59

Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Here we use Pearson correlation as similarity:
1) Subtract mean rating m_i from each movie ***i***
$m_1 = (1+3+5+5+4)/5 = 3.6$
row 1: *[-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]*
2) Compute cosine similarities between rows

Item-Item CF (INI=2)

users

movies

	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		**0.59**

Compute similarity weights:

$s_{1,3}=0.41$, $s_{1,6}=0.59$

Item-Item CF (INI=2)

users

movies

	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		2.6	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Predict by taking weighted average:

$$r_{ix} = \sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}$$

$r_{1.5}$ = **(0.41*2 + 0.59*3) / (0.41+0.59) = 2.6**

CF: Common Practice

Before:

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

- Define **similarity** s_{ij} of items ***i*** and ***j***
- Select ***k*** nearest neighbors $N(i; x)$
  - Items most similar to ***i***, that were rated by ***x***
- Estimate rating r_{xi} as the weighted average:

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

baseline estimate for r_{xi}

$b_{xi} = \mu + b_x + b_i$

- μ = overall mean movie rating
- b_x = rating deviation of user ***x***
  = (*avg. rating of user* ***x***) – μ
- b_i = rating deviation of movie ***i***

Item-Item vs. User-User

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.8	
Bob		0.5		0.3
Carol	0.9		1	0.8
David			1	0.4

- In practice, it has been observed that item-item often works better than user-user
  - Why? Items are simpler, users have multiple tastes

Pros/Cons of Collaborative Filtering

- **+ Works for any kind of item**
  - No feature selection needed
- **- Cold Start:**
  - Need enough users in the system to find a match
- **- Sparsity:**
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items
- **- First rater:**
  - Cannot recommend an item that has not been previously rated
  - New items, Esoteric items
- **- Popularity bias:**
  - Cannot recommend items to someone with unique taste
  - Tends to recommend popular items

Collaborative Filtering: Complexity

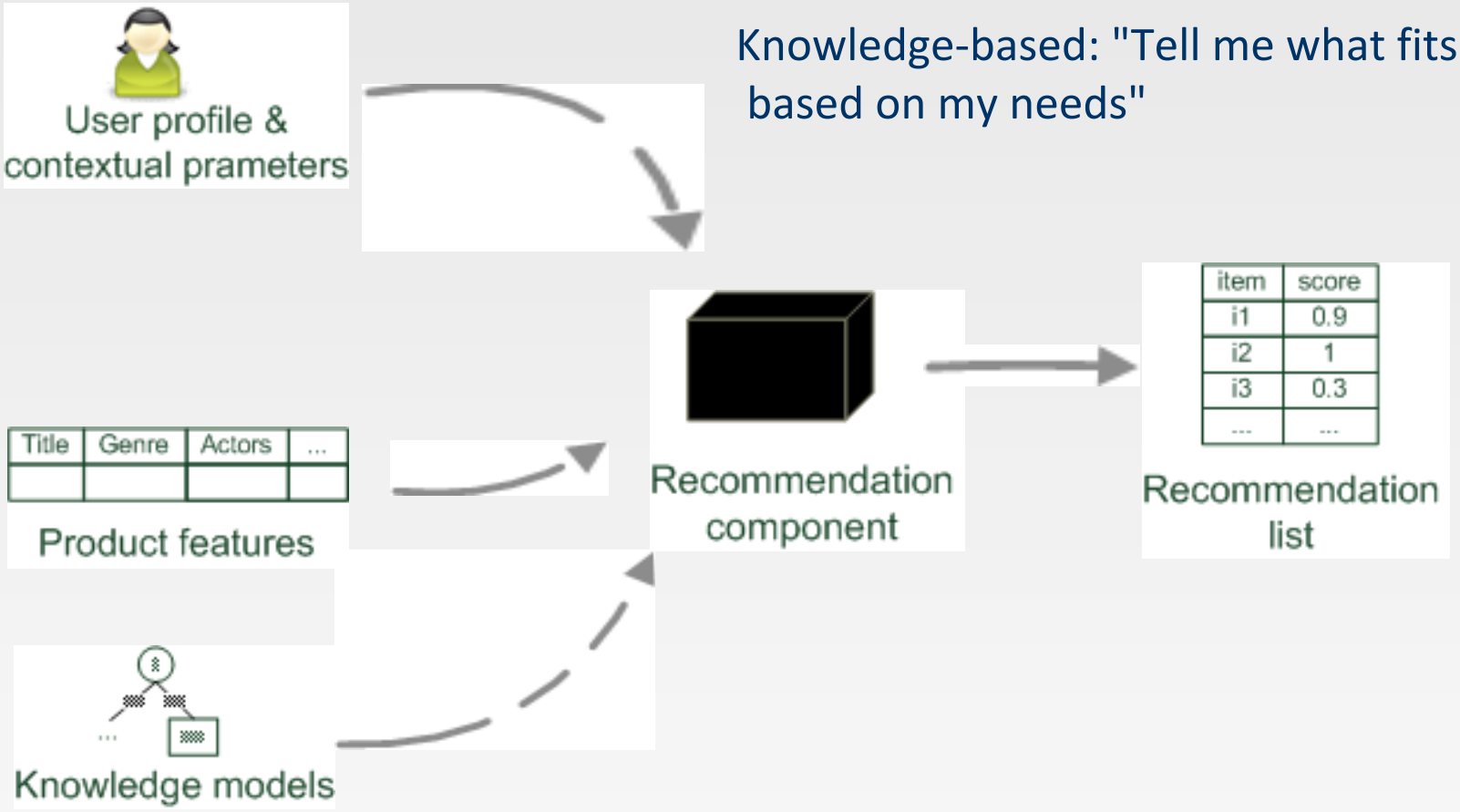
- Expensive step is finding ***k*** most similar customers: **O(|X|)**
- **Too expensive to do at runtime**
  - Could pre-compute
- Naïve pre-computation takes time **O(k ·|X|)**
  - X … set of customers
- Ways of doing this:
  - Near-neighbor search in high dimensions (**LSH**)
  - Clustering
  - Dimensionality reduction
  - … …
- Supported by Hadoop: Apache Mahout
  https://mahout.apache.org/users/basics/algorithms.html

Why do we need knowledge-based recommendation?

- Products with low number of available ratings

- Time span plays an important role
  - Five-year-old ratings for computers
  - User lifestyle or family situation changes
- Customers want to define their requirements explicitly
  - “The color of the car should be black"

Knowledge-Based Recommendation



Knowledge-based Recommendation

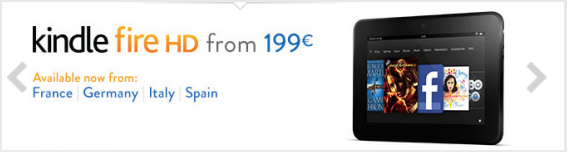
- Constraint-based
  - based on explicitly defined set of recommendation rules
  - fulfill recommendation rules
- Case-based
  - based on different types of similarity measures
  - retrieve items that are similar to specified requirements
- Both approaches are similar in their conversational recommendation process
  - users specify the requirements
  - systems try to identify solutions
  - if no solution can be found, users change requirements

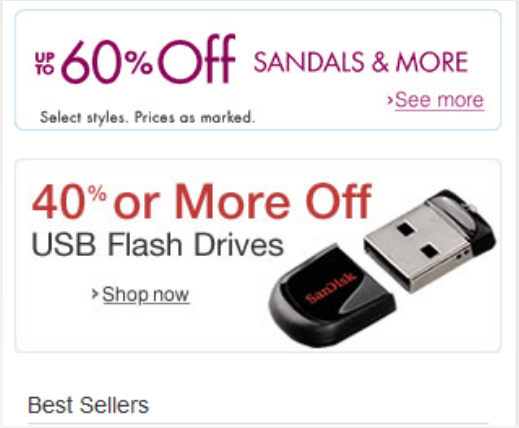
Hybrid Methods

- Implement two or more different recommenders and combine predictions
  - Perhaps using a linear model
- Add content-based methods to collaborative filtering
  - Item profiles for new item problem
  - Demographics to deal with new user problem

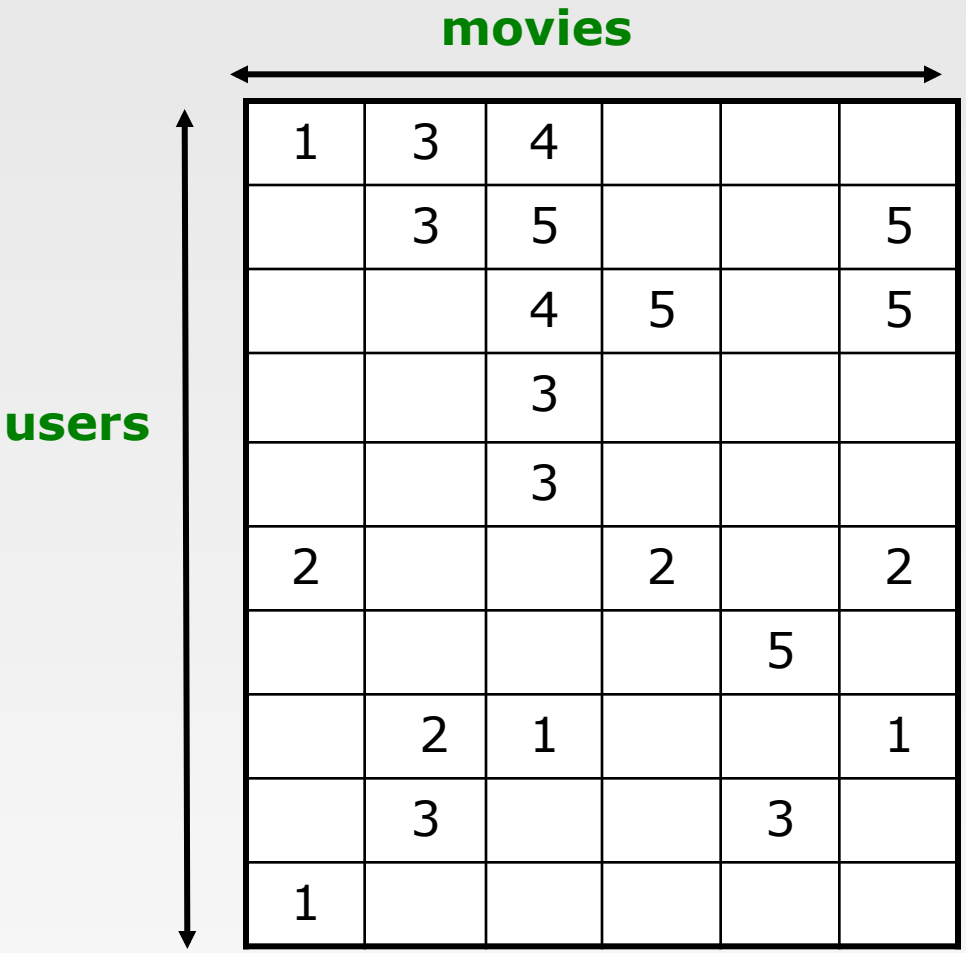
What is a Good Recommendation in Practice?

- Total sales numbers
- Promotion of certain items
- …
- Click-through-rates
- Interactivity on platform
- …
- Customer return rates
- Customer satisfaction and loyalty



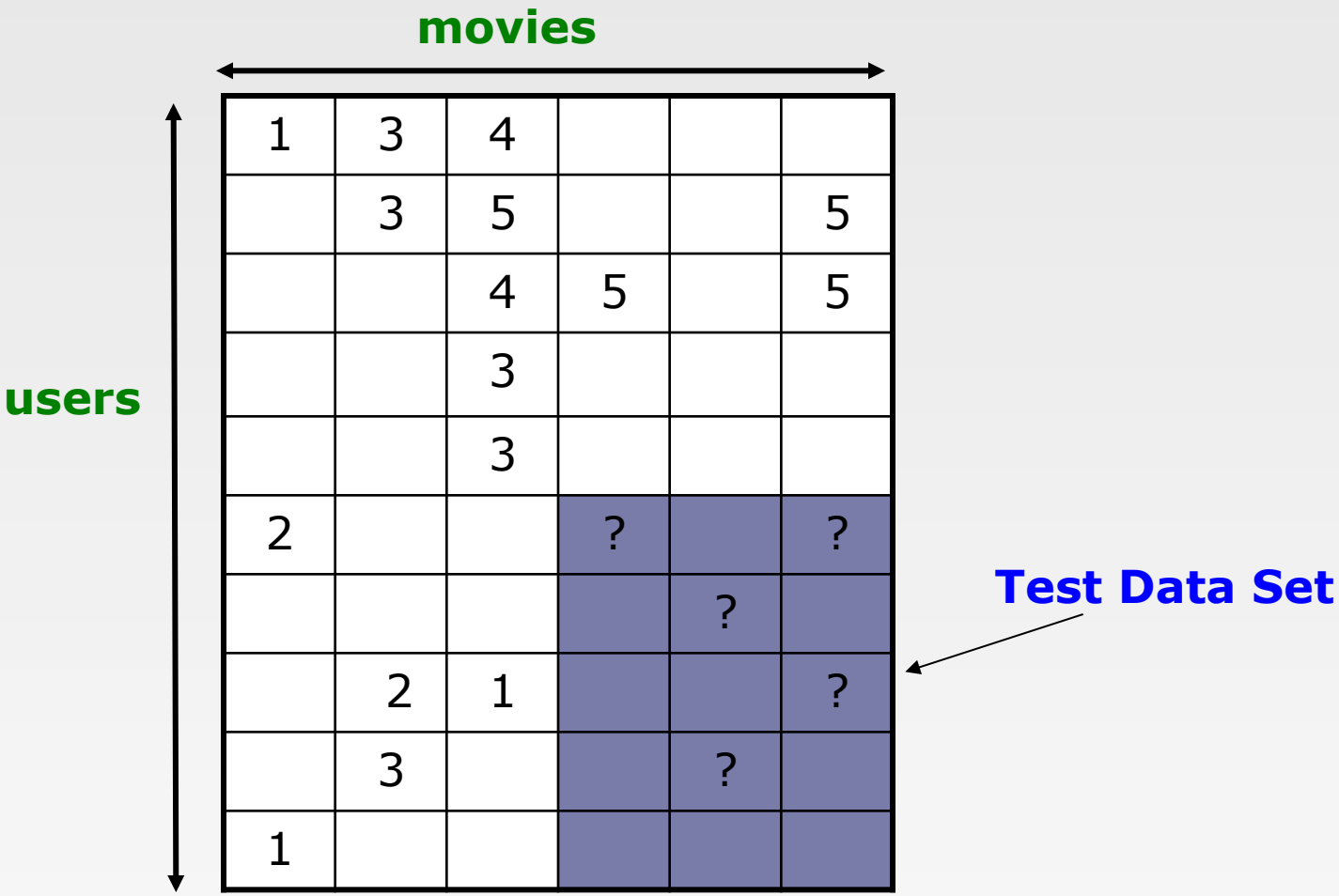


Evaluation



1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

Evaluation



Evaluating Predictions

- **Compare predictions with known ratings**
  - **Root-mean-square error** (RMSE)
    - $\sqrt{\square\sum xi\uparrow\blacksquare(r\downarrow xi-r\downarrow xi\uparrow*)\uparrow 2}$ where $\boldsymbol{rxi}$ is predicted, $\boldsymbol{r\downarrow xi\uparrow*}$ is the true rating of ***x*** on ***i***
  - **Precision at top 10**:
    - % of those in top 10
  - **Rank Correlation**:
    - Spearman's *correlation* between system's and user's complete rankings
- **Another approach: 0/1 model**
  - **Coverage:**
    - Number of items/users for which system can make predictions
  - **Precision:**
    - Accuracy of predictions
  - **Receiver operating characteristic** (ROC)
    - Tradeoff curve between false positives and false negatives

References

- Chapter 9 of Mining of Massive Datasets.
- Tutorial: Recommender Systems. IJCAI 2013.

End of Chapter 11