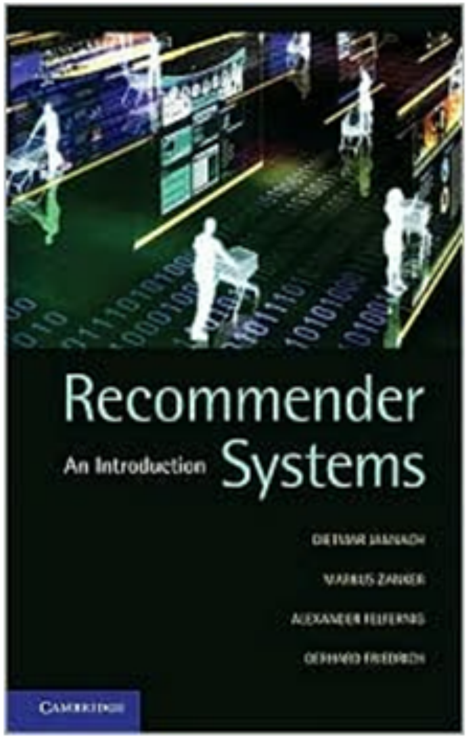
Recommender Systems

Instructor: Junghye Lee

Department of Industrial Engineering
junghyelee@unist.ac.kr



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

Contents

- **What are recommender systems for?**
  - Introduction
- **How do they work (Part I) ?**
  - Collaborative Filtering
- **How do they work (Part II) ?**
  - Content-based Filtering
  - Knowledge-Based Recommendations
- **How to measure their success?**
  - Evaluation techniques

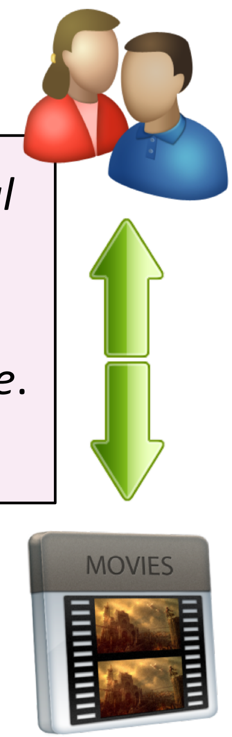
Introduction

Problem domain

- **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, MISQ, 2007]



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

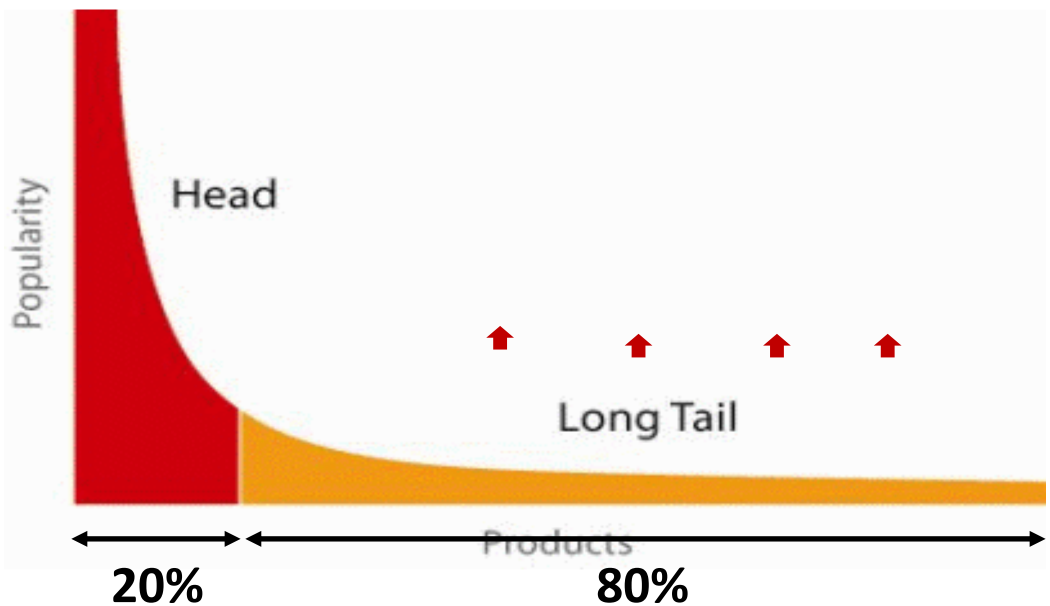
Purpose and success criteria (1)

- **Different perspectives/aspects**
  - Depends on domain and purpose
  - No holistic evaluation scenario exists
- **Retrieval perspective**
  - Reduce search costs
  - Provide "correct" proposals
  - Users know in advance what they want
- **Recommendation perspective**
  - Serendipity – identify items from the Long Tail
  - Users did not know about existence

When does a RS do its job well?

"Recommend widely unknown items that users might actually like!"

Recommend items from the long tail



20% of items accumulate 74% of all positive ratings

Items rated > 3 in MovieLens 100K dataset

Purpose and success criteria (2)

- **Prediction perspective**
  - Predict to what degree users like an item
  - Most popular evaluation scenario in research
- **Interaction perspective**
  - Give users a "good feeling"
  - Educate users about the product domain
  - Convince/persuade users - explain
- **Finally, conversion perspective**
  - Commercial situations
  - Increase "hit", "clickthrough", "lookers to bookers" rates
  - Optimize sales margins and profit

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.

- **Relation to Information Retrieval:**
  - IR is finding material [..] of an unstructured nature [..] that satisfies an information need from within large collections [..].
    - [Manning et al., CUP, 2008]

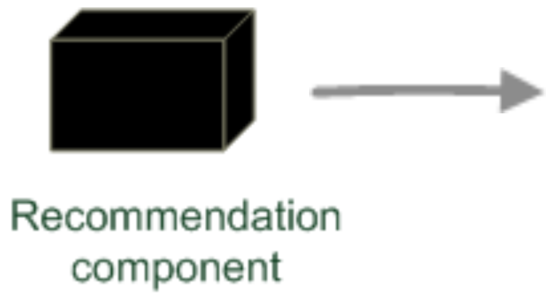
Recommender systems

- **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.

- **Relation to Information Retrieval:**
  - IR is finding material [..] of an unstructured nature [..] that satisfies an information need from within large collections [..].
    - » [Manning et al., CUP, 2008]

Paradigms of recommender systems

Recommender systems reduce information overload by estimating relevance

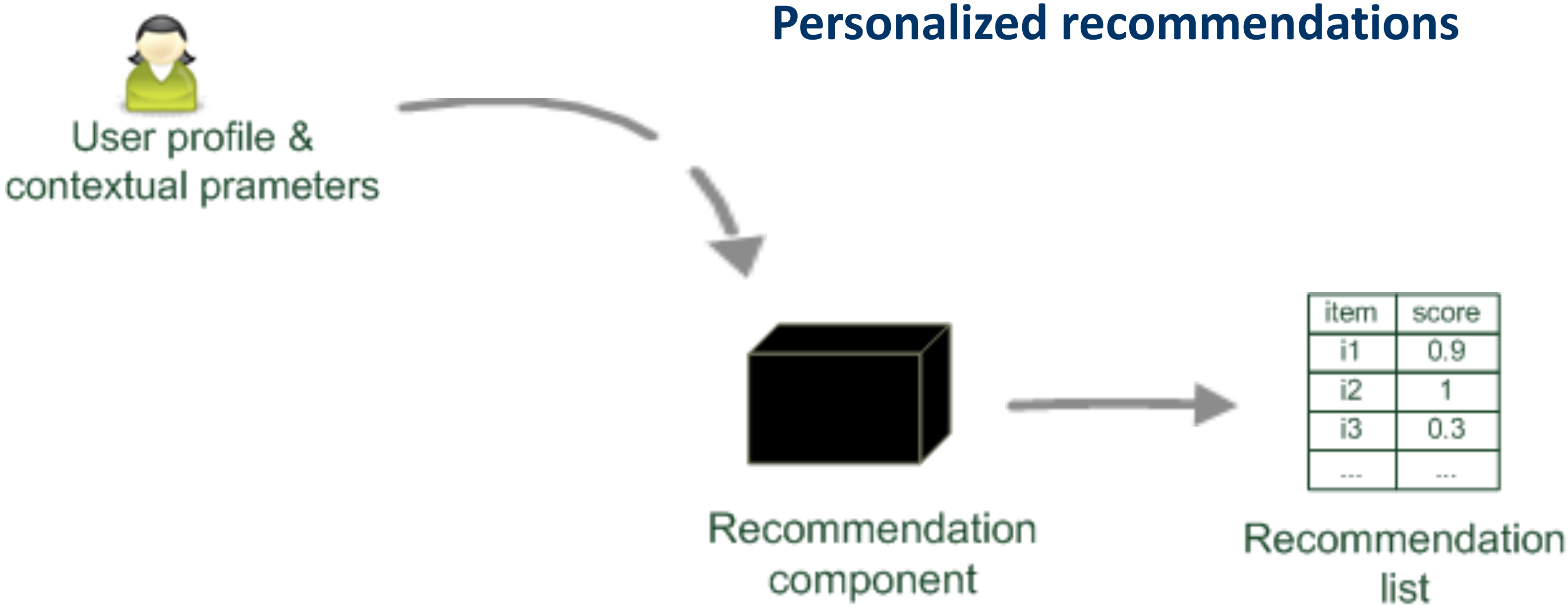


item	score
i1	0.9
i2	1
i3	0.3
...	...

Recommendation
list

Paradigms of recommender systems

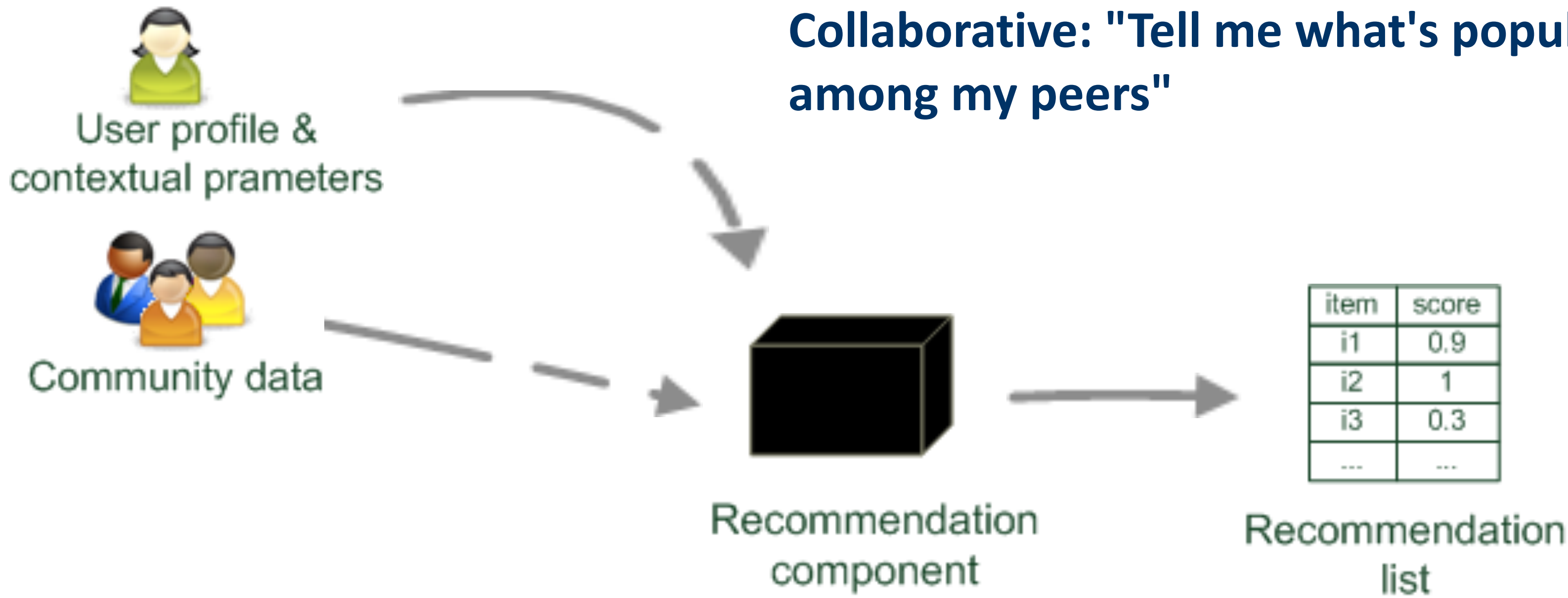
Personalized recommendations



item	score
i1	0.9
i2	1
i3	0.3
...	...

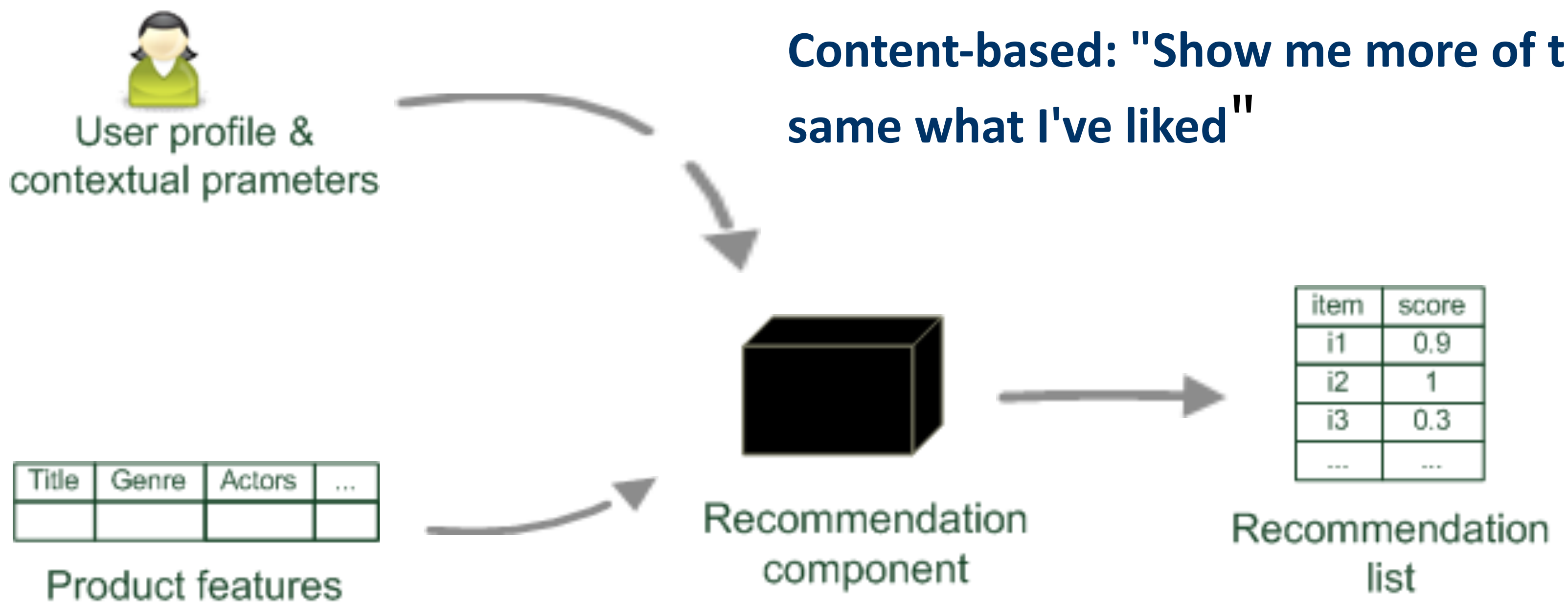
Paradigms of recommender systems

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



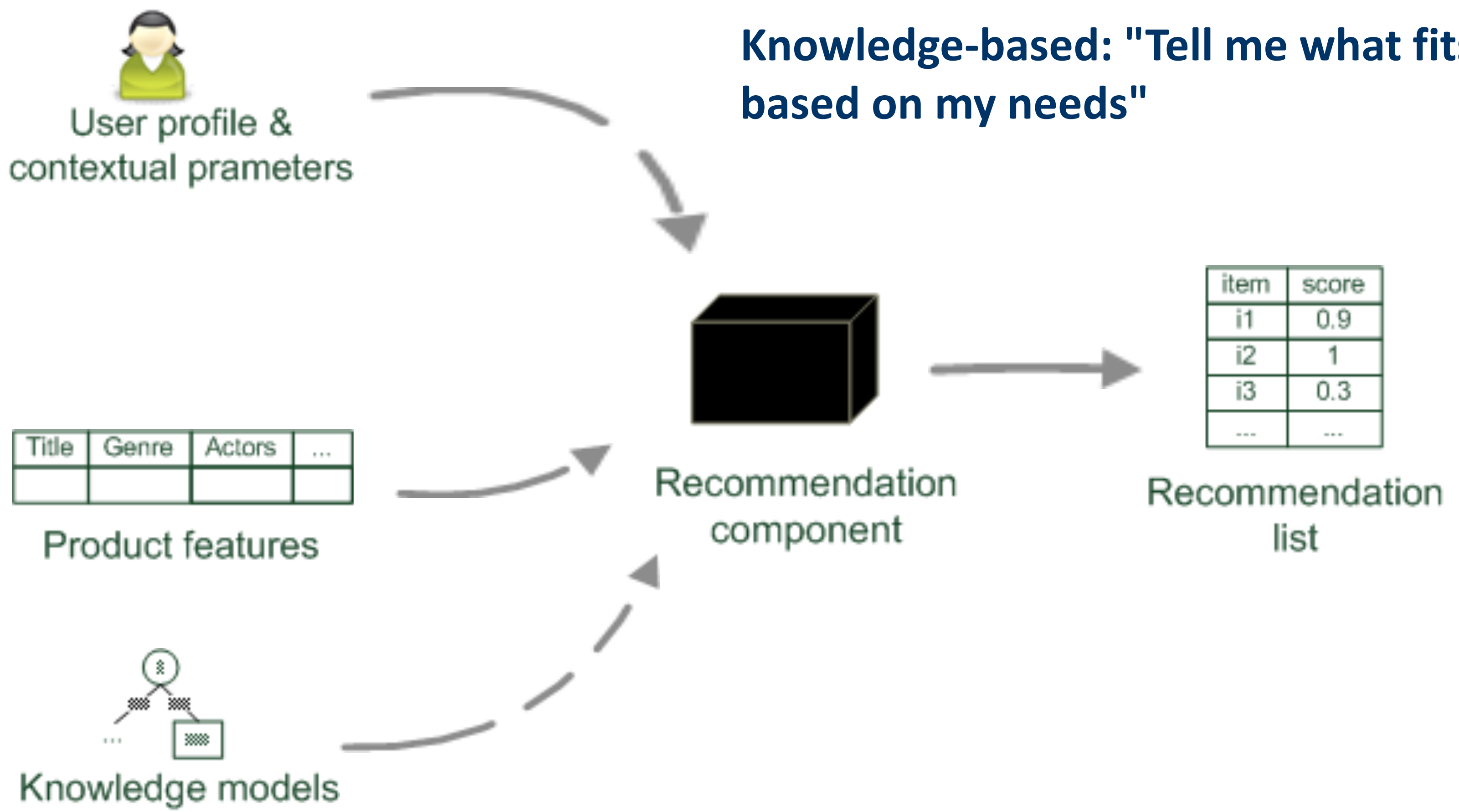
Paradigms of recommender systems

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



Paradigms of recommender systems

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



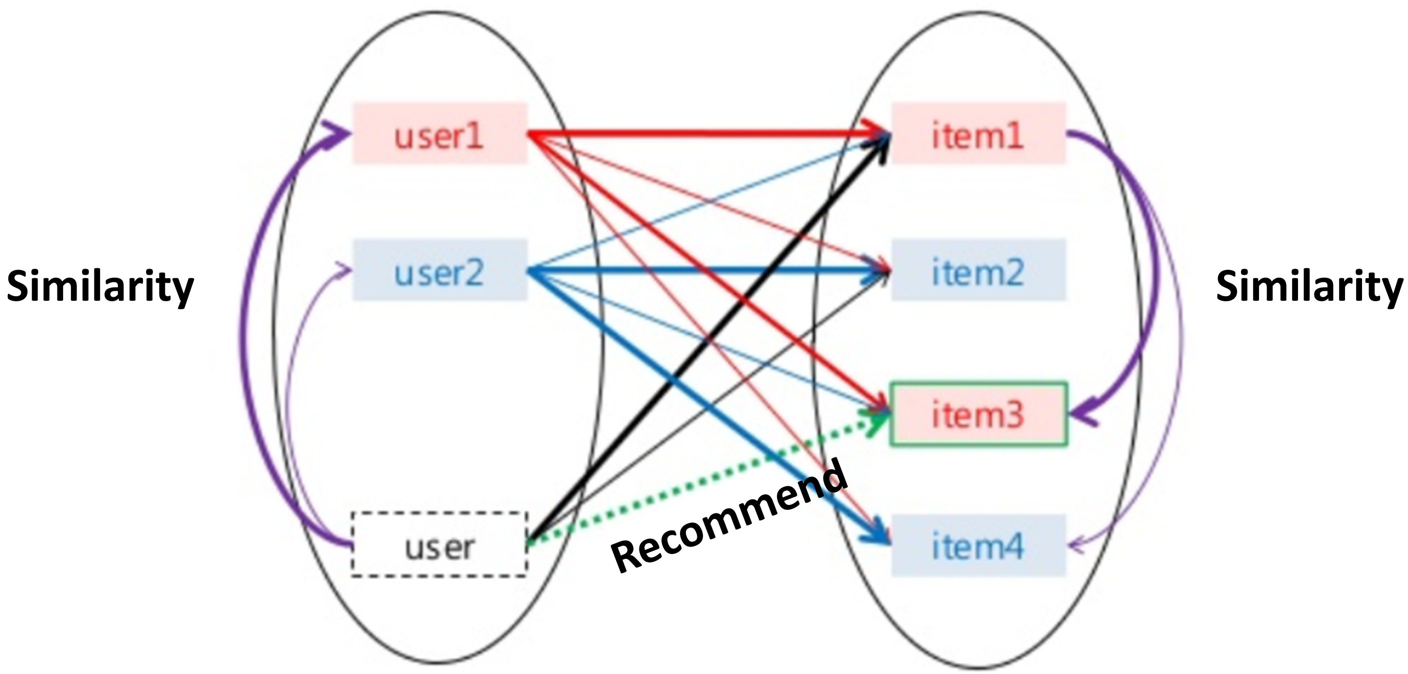
Collaborative Filtering

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

User-based vs. Item-based CF

- **The way to recommend either item 3 and item 4 to user**
  - user is similar to user 1, so recommend item 3 which is preferred by user 1
    - In terms of USER
  - user prefers item 1 and item 1 is similar to item 3, so recommend item 3
    - In terms of ITEM



User-based nearest-neighbor collaborative filtering (1)

- **The basic technique:**
  - Given an "active user" (Alice) and an item *p* 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 *p*
    - use, e.g. the average of their ratings to predict, if Alice will like item *p*
    - 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 (2)

- **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

- **A popular similarity measure in user-based CF**

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

a, b : users

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

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

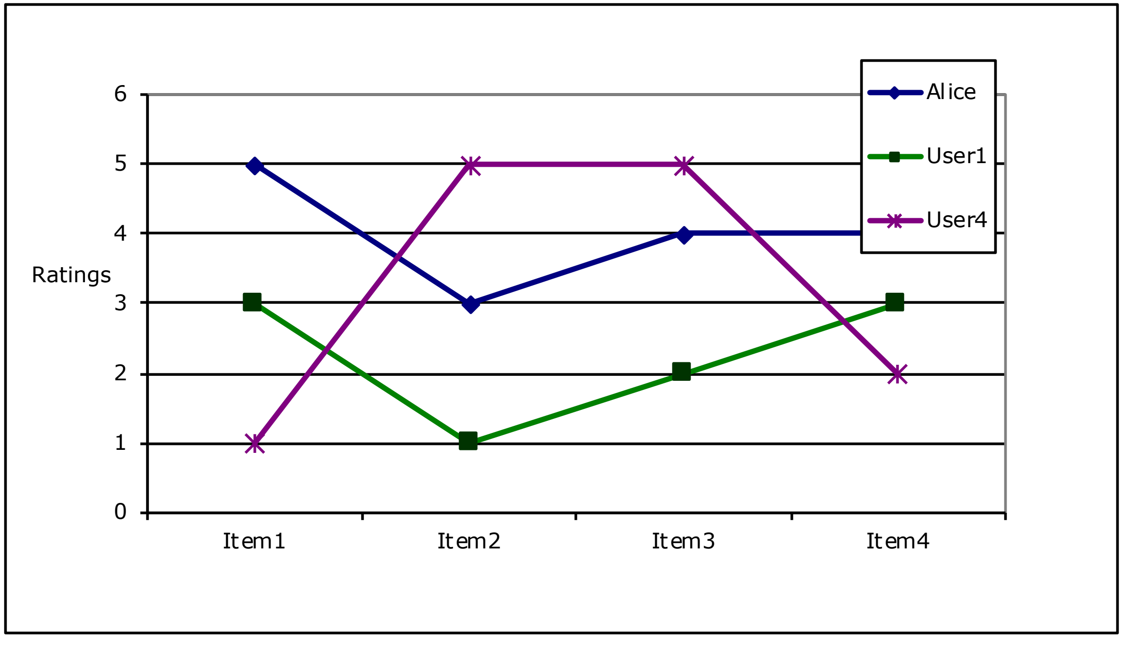
$\bar{r}_a, \bar{r}_b$: user's average ratings

	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

sim = 0.85
sim = 0.70
sim = -0.79

Pearson correlation

- **Takes differences in rating behavior into account**



- **Works well in usual domains, compared with alternative measures**
  – such as cosine similarity

Making predictions

- A common prediction function:

$$\boldsymbol{pred(a,p) = \bar{r}_a + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \bar{r}_b)}{\sum_{b \in N} |sim(a,b)|}}$$

where user *a* and item *p, N*: the number of users

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

Improving the metrics / prediction function

- **Not all neighbor ratings might be equally "valuable"**
  - Agreement on commonly liked items is not so informative as agreement on controversial items
  - **Possible solution**: Give more weight to items that have a higher variance
- **Value of number of co-rated items**
  - Use "significance weighting", by e.g., linearly reducing the weight when the number of co-rated items is low
- **Case amplification**
  - Intuition: Give more weight to "very similar" neighbors, i.e., where the similarity value is close to 1.
- **Neighborhood selection**
  - Use similarity threshold or fixed number of neighbors

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

2001: *Item-based collaborative filtering recommendation algorithms*, B. Sarwar et al., WWW 2001

- **Scalability issues arise with U2U if many more users than items (m >> n , m = |users|, n = |items|)**
  - e.g. amazon.com
  - Space complexity $O(m^2)$ when pre-computed
  - Time complexity for computing Pearson $O(m^2 n)$

- **High sparsity leads to few common ratings between two users**

- **Basic idea: "Item-based CF exploits relationships between items first, instead of relationships between users"**

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

Similarity measure

- **Produces better results in item-to-item filtering**
  - for some datasets, no consistent picture in literature
- **Ratings are seen as vector in n-dimensional space**
- **Cosine similarity**

$$sim(\vec{p}, \vec{q}) = \frac{\vec{p} \cdot \vec{q}}{|\vec{p}| * |\vec{q}|}$$

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

$$sim(p, q) = \frac{\sum_{u \in U} \left(r_{u,p} - \bar{r}_p\right)\left(r_{u,q} - \bar{r}_q\right)}{\sqrt{\sum_{u \in U} \left(r_{u,p} - \bar{r}_p\right)^2} \sqrt{\sum_{u \in U} \left(r_{u,q} - \bar{r}_q\right)^2}}$$

Similarity measure

$$pred(a,p) = \frac{\sum_{q \in S(p,a,k)} sim(p,q) * r_{a,q}}{\sum_{q \in S(p,a,k)} |sim(p,q)|}$$

- **Produces better results in item-to-item filtering**
  - for some datasets, no consistent picture in literature
- **Ratings are seen as vector in n-dimensional space**
- **Cosine similarity**

$$sim(\vec{p},\vec{q}) = \frac{\vec{p}\cdot\vec{q}}{|\vec{p}| * |\vec{q}|}$$

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

$$sim(p,q) = \frac{\sum_{u \in U} (r_{u,p} - \bar{r}_p)(r_{u,q} - \bar{r}_q)}{\sqrt{\sum_{u \in U} (r_{u,p} - \bar{r}_p)^2} \sqrt{\sum_{u \in U} (r_{u,q} - \bar{r}_q)^2}}$$

Pre-processing for item-based filtering

- **Item-based filtering does not solve the scalability problem itself**
- **Pre-processing approach by Amazon.com (in 2003)**
  - Calculate all pair-wise item similarities in advance
  - The neighborhood to be used at run-time is typically rather small, because only items are taken into account which the user has rated
  - Item similarities are supposed to be more stable than user similarities
- **Memory requirements**
  - Up to n^2 pair-wise similarities to be memorized (n = number of items) in theory
  - In practice, this is significantly lower (items with no co-ratings)
  - Further reductions possible
    - Minimum threshold for co-ratings (items, which are rated at least by *l* users)
    - Limit the size of the neighborhood (*k* might affect recommendation accuracy)

More on ratings

- **Pure CF-based systems only rely on the rating matrix**
- **Explicit 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 domain
    - Multidimensional ratings (multiple ratings per movie)
  - Challenge
    - Users not always willing to rate many items; sparse rating matrices
    - How to stimulate users to rate more items?
- **Implicit ratings**
  - clicks, page views, time spent on some page, demo downloads, and etc. 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
- **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

Example algorithms for sparse datasets

- **Recursive CF**
  - Assume there is a very close neighbor *b* of *a* who however has not rated the target item *p* yet.
  - Idea:
    - Apply CF-method recursively and predict a rating for item *p* for the neighbor
    - Use this predicted rating instead of the rating of a more distant direct neighbor

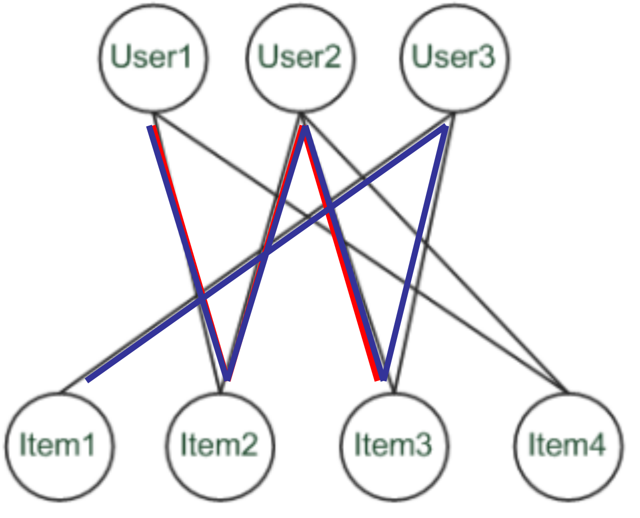
	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	?
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = 0.85

Predict rating for User1

Graph-based methods

- **"Spreading activation" (sketch)**
  - Idea: Use paths of lengths > 3 to recommend items
  - Length 3: Recommend Item3 to User1
  - Length 5: Item1 also recommendable



Bipartite graph, odd number!

More model-based approaches

- **Plethora of different techniques proposed in the last years, e.g.,**
  - Matrix factorization techniques, statistics
    - singular value decomposition, principal component analysis
  - Association rule mining
    - compare: shopping basket analysis
  - Probabilistic models
    - clustering models, Bayesian networks, probabilistic Latent Semantic Analysis
  - Various other machine learning approaches
- **Costs of pre-processing**
  - Usually not discussed
  - Incremental updates possible?

2000: *Application of Dimensionality Reduction in Recommender System*, B. Sarwar et al., WebKDD Workshop

- **Basic idea: Trade more complex offline model building for faster online prediction generation**
- **Singular Value Decomposition for dimensionality reduction of rating matrices**
  - Captures important factors/aspects and their weights in the data
  - factors can be genre, actors but also non-understandable ones
  - Assumption that k dimensions capture the signals and filter out noise (K = 20 to 100)
- **Constant time to make recommendations**
- **Approach also popular in IR (Latent Semantic Indexing), data compression,…**

Matrix factorization

- **SVD:** $$\boldsymbol{M_k = U_k \times \Sigma_k \times V_k^T}$$

U_k	Dim1	Dim2
Alice	0.47	-0.30
Bob	-0.44	0.23
Mary	0.70	-0.06
Sue	0.31	0.93

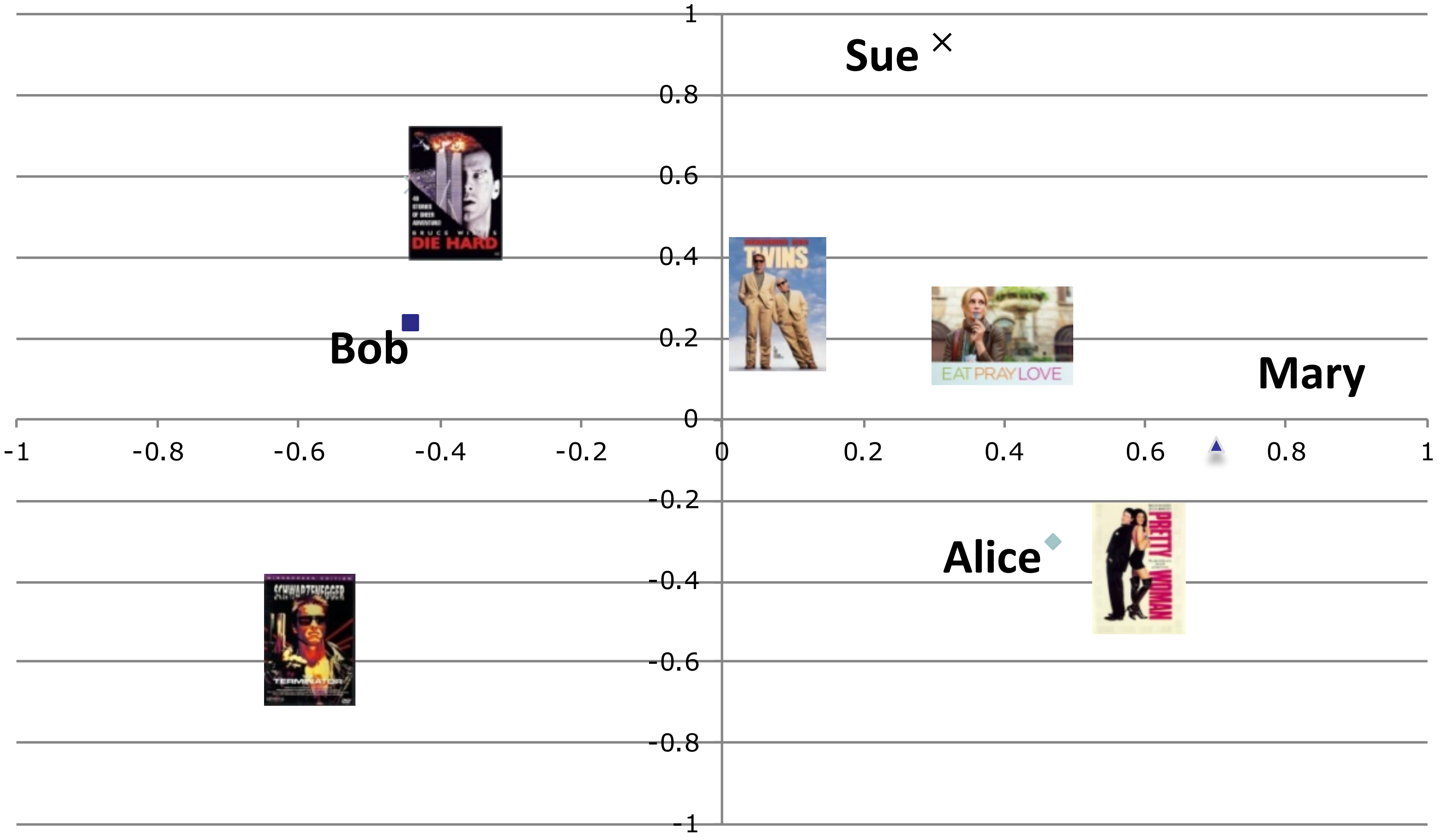
V_k^T					
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

Σ_k	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

- **Prediction:** $\hat{r}_{ui} = \bar{r}_u + U_k(Alice) \times \Sigma_k \times V_k^T(EPL)$

= 3 + 0.84 = 3.84

A picture says …



Association rule mining

- **Commonly used for shopping behavior analysis**
  - aims at detection of rules such as

    "If a customer purchases baby-food then he also buys diapers in 70% of the cases"

- **Association rule mining algorithms**
  - can detect rules of the form X => Y (e.g., baby-food => diapers) from a set of sales transactions D = {t_1, t_2, … t_n}
  - measure of quality: support, confidence
    - used e.g. as a threshold to cut off unimportant rules
  - let $\sigma(X) = \frac{|\{x|x \subseteq t_1, t_1 \in D\}|}{|D|}$
  - support = $\frac{\sigma(X \cup Y)}{|D|}$, confidence = $\frac{\sigma(X \cup Y)}{\sigma(X)}$

$$support = \frac{|X \cup Y|}{|Transactions|}$$

$$confidence = \frac{|X \cup Y|}{|X|}$$

Recommendation based on Association Rule Mining

	Item1	Item2	Item3	Item4	Item5
Alice	1	0	0	0	?
User1	1	0	1	0	1
User2	1	0	1	0	1
User3	0	0	0	1	1
User4	0	1	1	0	0

- **Simplest approach**
  - transform 5-point ratings into binary ratings (1 = above user average)
- **Mine rules such as**
  - Item1 => Item5
    - support (2/4), confidence (2/2) (without Alice)
- **Make recommendations for Alice (basic method)**
  - Determine "relevant" rules based on Alice's transactions (the above rule will be relevant as Alice bought Item1)
  - Determine items not already bought by Alice
  - Sort the items based on the rules' confidence values
- **Different variations possible**
  - dislike statements, user associations ..

Probabilistic methods

- **Basic idea (simplistic version for illustration):**
  - given the user/item rating matrix
  - determine the probability that user Alice will like an item *p*
  - base the recommendation on such these probabilities
- **Calculation of rating probabilities based on Bayes Theorem**
  - How probable is rating value "1" for Item5 given Alice's previous ratings?
  - Corresponds to conditional probability *P*(Item5=1|*X*), where
    - *X* = Alice's previous ratings = (Item1 =1, Item2=3, Item3= …)
  - Can be estimated based on Bayes Theorem

$$P(Y|X) = \frac{P(X|Y) \times P(Y)}{P(X)} \qquad P(Y|X) = \frac{\prod_{i=1}^{d} P(X_i|Y) \times P(Y)}{P(X)}$$

  - Assumption: Ratings are independent, *d* = number of attributes in *X*, # of items

Calculation of probabilities in simplistic approach

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

X = (Item1 = 1, Item2 = 3, Item3 = …)

$$\begin{aligned}
P(X|Item5=1) &= P(Item1=1|Item5=1) \times P(Item2=3|Item5=1) \times \\
&\quad P(Item3=3|Item5=1) \times P(Item4=2|Item5=1) \\
&= 2/4 \times 1/4 \times 1/4 \times 1/4 \\
&\approx 0.0078125 \\
P(X|Item5=2) &= P(Item1=1|Item5=2) \times P(Item2=3|Item5=2) \times \\
&\quad P(Item3=3|Item5=2) \times P(Item4=2|Item5=2) \\
&= 0/4 \times \ldots \times \ldots \times \ldots. \\
&= 0
\end{aligned}$$

- **More to consider**
  - Zeros (smoothing required)
  - like/dislike simplification possible

Practical probabilistic approaches

- **Use a cluster-based approach**
  - assume users fall in a small number of subgroups (clusters)
  - Make predictions based on estimates
    - probability of Alice falling into cluster *c*
    - probability of Alice liking item *i* given a certain cluster and her previous ratings
  - Based on model-based clustering (mixture model)
    - Number of classes and model parameters have to be learned from data in advance (EM algorithm)
- **Others:**
  - Bayesian Networks, Probabilistic Latent Semantic Analysis, ...
- **Empirical analysis shows:**
  - Probabilistic methods lead to relatively good results (movie domain)
  - No consistent winner; small memory-footprint of network model

Collaborative Filtering Issues

- **Pros:**
  - well-understood, works well in some domains, no knowledge engineering required
- **Cons:**
  - requires user community, sparsity problems, no integration of other knowledge sources, no explanation of results
- **What is the best CF method?**
  - In which situation and which domain? Inconsistent findings; always the same domains and data sets; differences between methods are often very small (1/100)
- **How to evaluate the prediction quality?**
  - MAE / RMSE: What does an MAE of 0.7 actually mean?
  - Serendipity: Not yet fully understood
- **What about multi-dimensional ratings?**