

**COMP9321**

# **Data Services Engineering**

**Term 1, 2019**

**Week 8 Lecture 1**

# Quiz 5

1. Which of the following statements is true for k-NN classifiers

- The classification accuracy is better with larger values of  $k$
- The decision boundary is smoother with smaller values of  $k$
- **k-NN does not require an explicit training step**
- The decision boundary is linear

# Quiz 5

2. What is the best approach to solve this question:

*What is the average weekly salary of all female employees under forty years of age?*

- Supervised learning
- Unsupervised learning
- Data query

# Quiz 5

3. A company has build a kNN classifier that gets 100% accuracy on training data. When they deployed this model on client side it has been found that the model is not at all accurate.

Which of the followings could be the reason:

- It is probably a overfitted model
- It is probably a underfitted model
- None of these

# Quiz 5

4. Considering the following training set of  $m = 4$  training examples:

Consider the linear regression model  $h_{\theta}(x) = \theta_0 + \theta_1 x$ .

What are the values of  $\theta_0$  and  $\theta_1$  that you would expect to obtain upon running gradient descent on this model? (Linear regression will be able to fit this data perfectly.)

x	y
1	0.5
2	1
4	2
0	0

- $\theta_0 = 1, \theta_1 = 0.5$
- $\theta_0 = 0, \theta_1 = 0.5$
- $\theta_0 = 0.5, \theta_1 = 0$
- $\theta_0 = 0.5, \theta_1 = 1$

# Quiz 5

3. For which of following tasks might K-means clustering be a suitable algorithm?

- Given historical weather records, predict if tomorrow's weather will be sunny or rainy
- Given a set of news articles from many different websites, find out what topics are the main topics covered
- Given sales data from a large number of products in a supermarket, estimate future sales for each of these products.

# Introduction to Recommender Systems

COMP9321 2019T1

# 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:

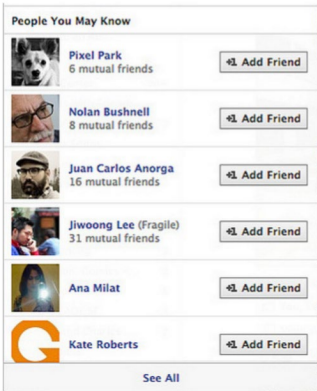
- Relevance score. Used for ranking.

Relation to Information Retrieval:

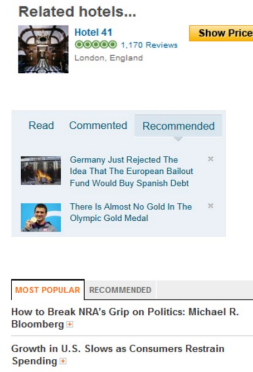
- IR is finding material ... of an unstructured nature ... that satisfies an information need from within large collections ... .

(Manning et al. 2008)





API/Software/App...

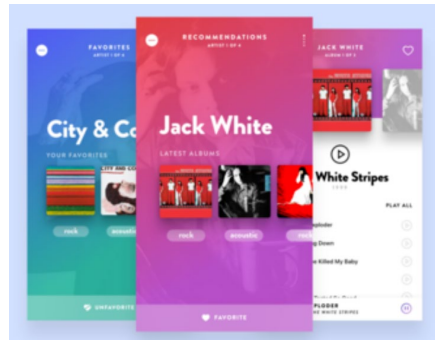
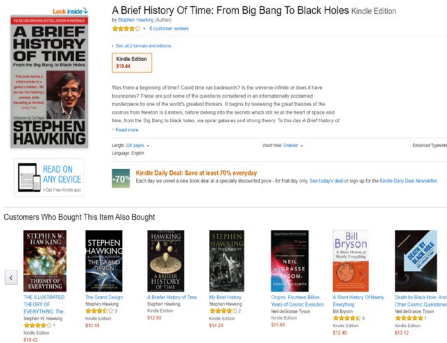


Tweet

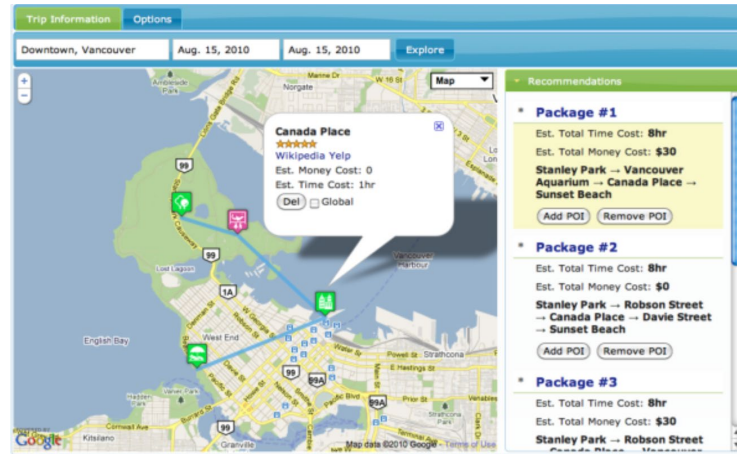
Hotel

Expert

People of Interest



Music



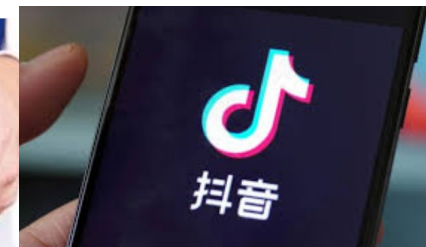
Trip

Book

Location/Route



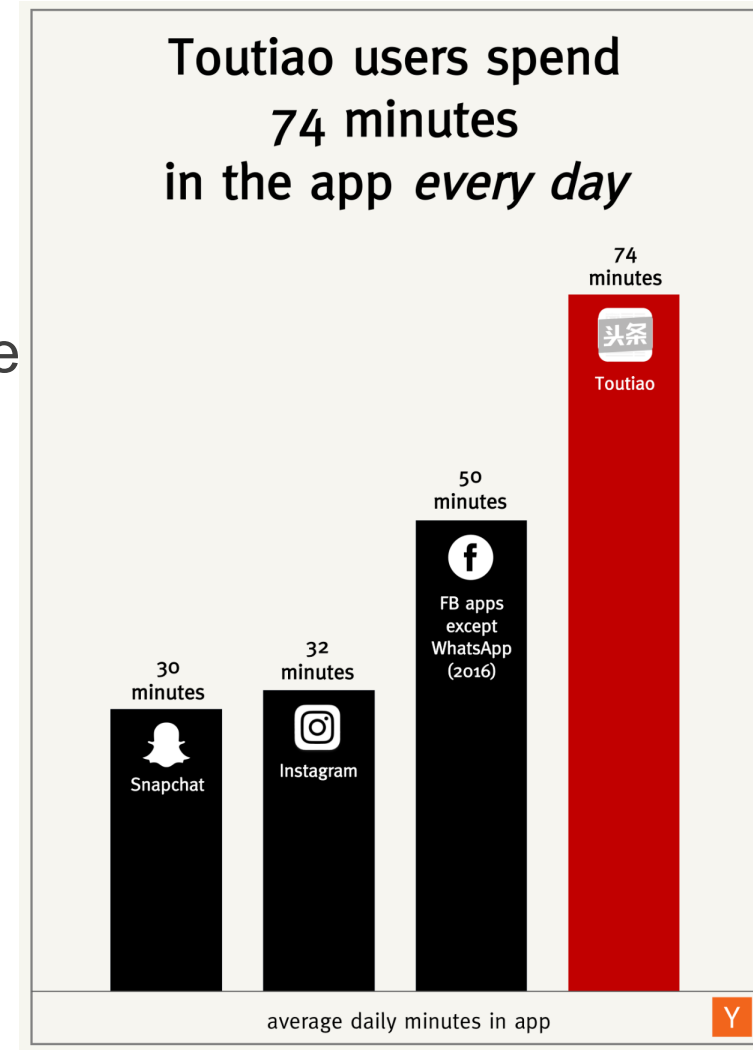
Medicine



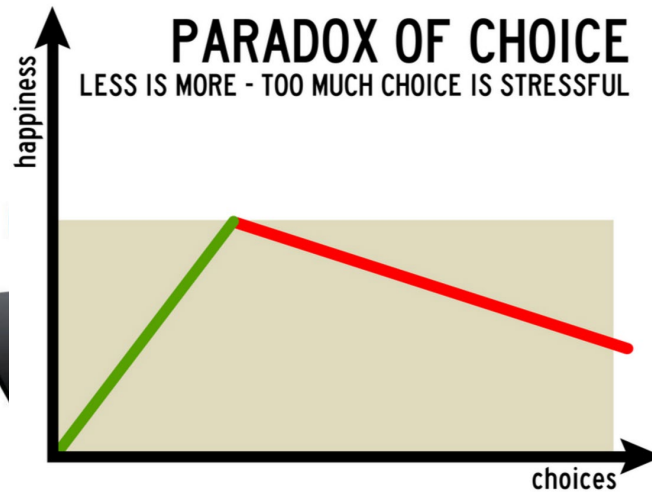
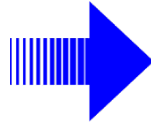
Video

## Why RS

- Netflix: 2/3 of the movies watched are recommended
- Google News: recommendations generate 38% more clickthrough
- Amazon: 35% sales from recommendations
- Toutiao: over 120M people in China use it each day (Nov 2017)



Sources: Snapchat – S-1 filing. Instagram – Recode. Facebook – Q1 2016 earnings report.



“People read around 10 MB worth of material a day, hear 400 MB a day, and see 1 MB of information every second”

— The Economist, November 2006

In 2015, consumption will raise to 74 GB a day - UCSD Study 2014

## Why RS

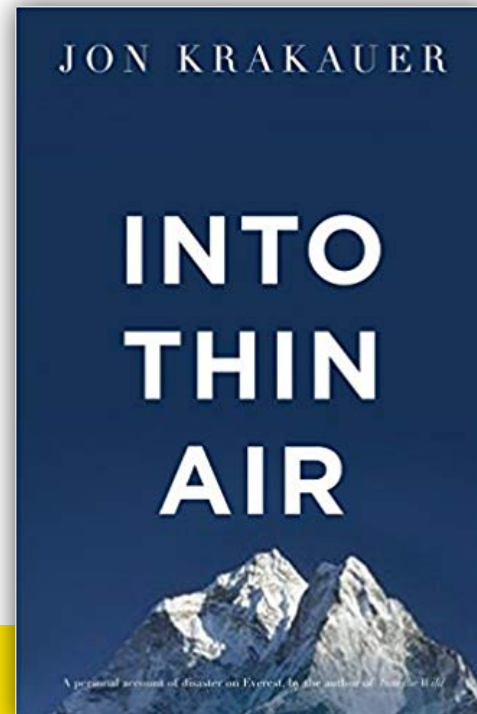
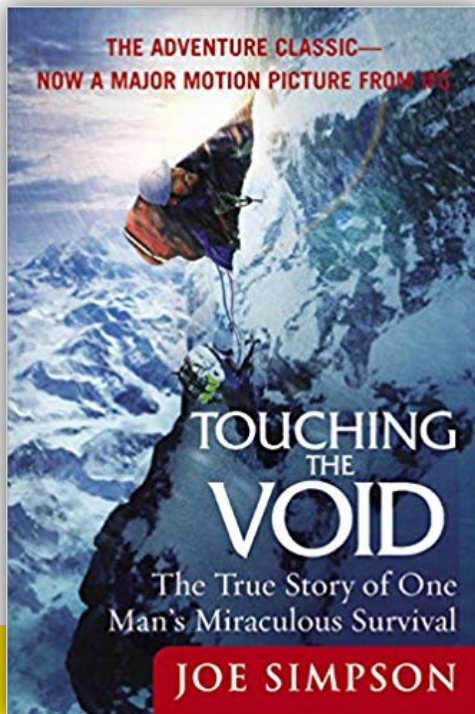
- *Recommend widely unknown items that users might actually like!"*
- *20% of items accumulate 74% of all positive ratings*

### ▪ Long Tail Phenomenon: scarcity-driven abundance



# The Long Tail

- Shelf space is a scarce commodity for traditional retailers
  - Also: TV networks, movie theaters,...
- The web enables near-zero-cost dissemination of information about products
- More choice necessitates better filters
  - Recommendation engines (e.g., Amazon)
  - How [Into Thin Air](#) made [Touching the Void](#) a bestseller



# Why RS?

- Value for the customers
  - Find things that are interesting
  - Narrow down the set of choices
  - Explore the space of options
  - Discover new things
  - ...
- Value for the providers
  - Boost profit ranging from 10% - 50% caused by accurate “You Might Also Like” recommendations
  - Improve retention, e.g., increase loyalty.
  - Guide/Change consuming behaviors
  - ...

# Search vs. Recommender Systems

**You don't need to look for the products and services, the products/services find you**

- Search allows you to search by entering the term, explicitly, while Recommender systems do not need a search term, it takes it implicitly
- Recommender system is proactive and a better user experience for finding things than traditional search

# Content

## Introduction

- Problem domain
- Purpose and success criteria
- Paradigms of recommender systems
  - Collaborative Filtering
  - Content-based Filtering
  - Knowledge-Based Recommendations
  - Hybridization Strategies



# 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 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!"
- 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

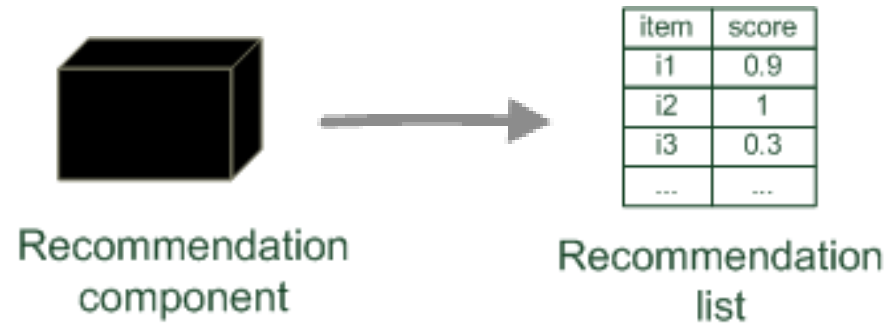
- Give users a "good feeling"
- Educate users about the product domain
- Convince/persuade users - explain

## Finally, conversion perspective

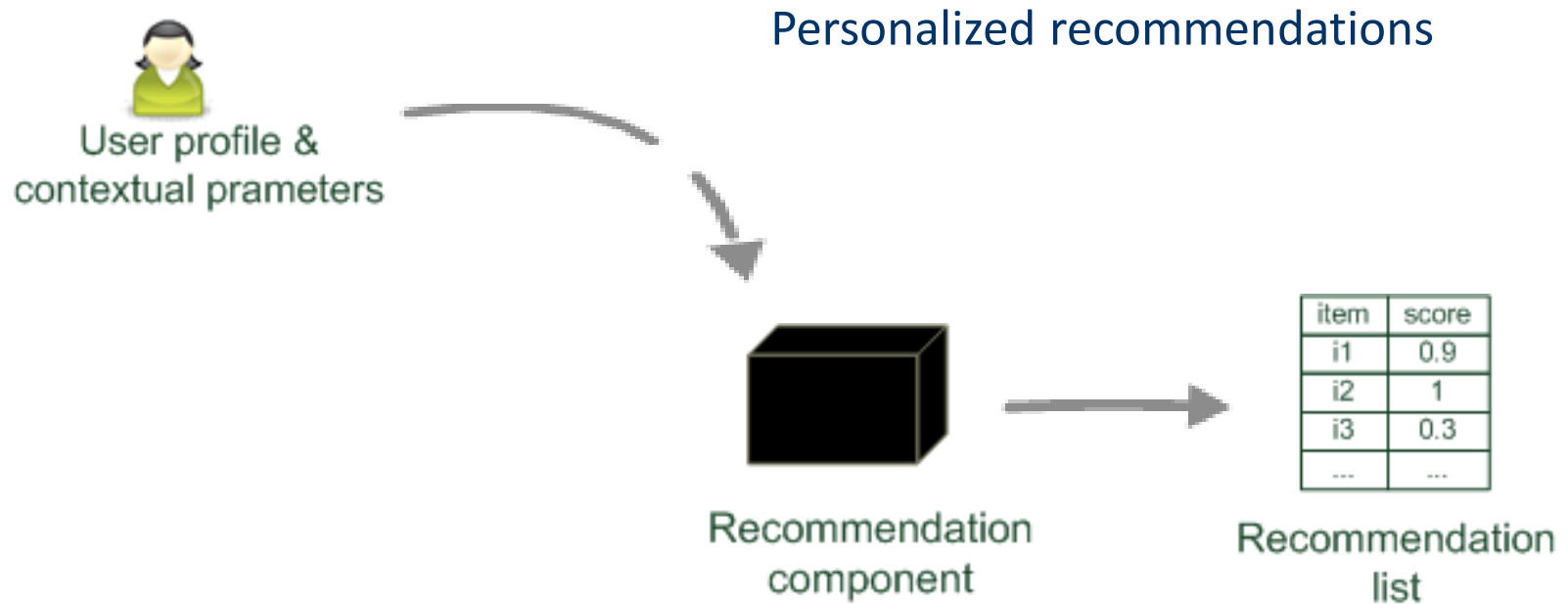
- Commercial situations
- Increase "hit", "clickthrough", "lookers to bookers" rates
- Optimize sales margins and profit

# Paradigms of recommender systems

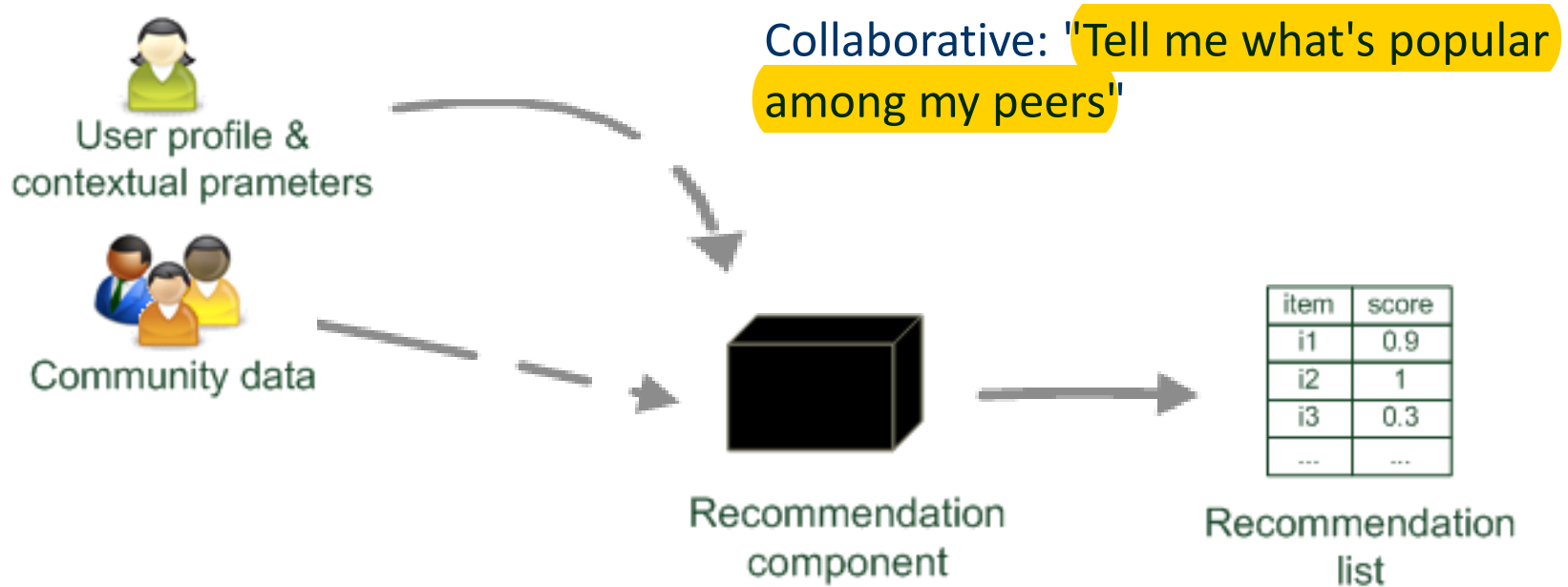
Recommender systems reduce information overload by estimating relevance



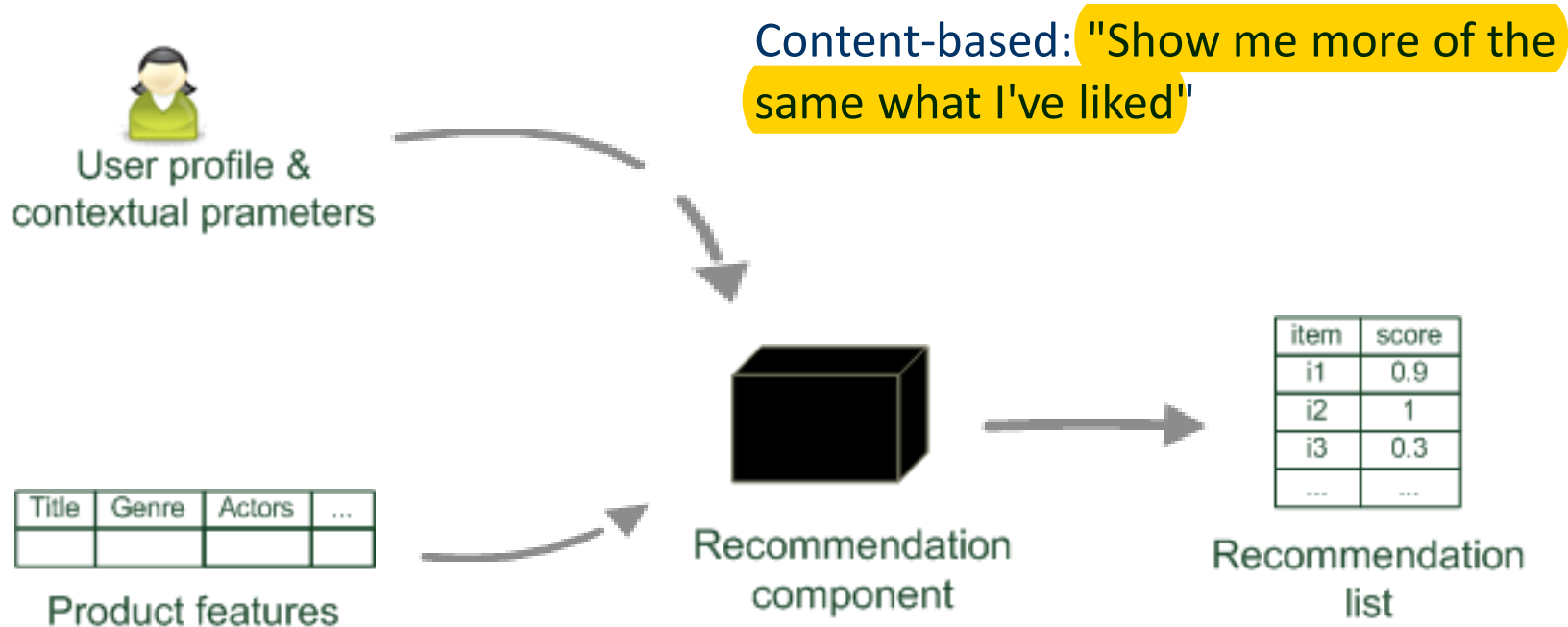
# Paradigms of recommender systems



# Paradigms of recommender systems

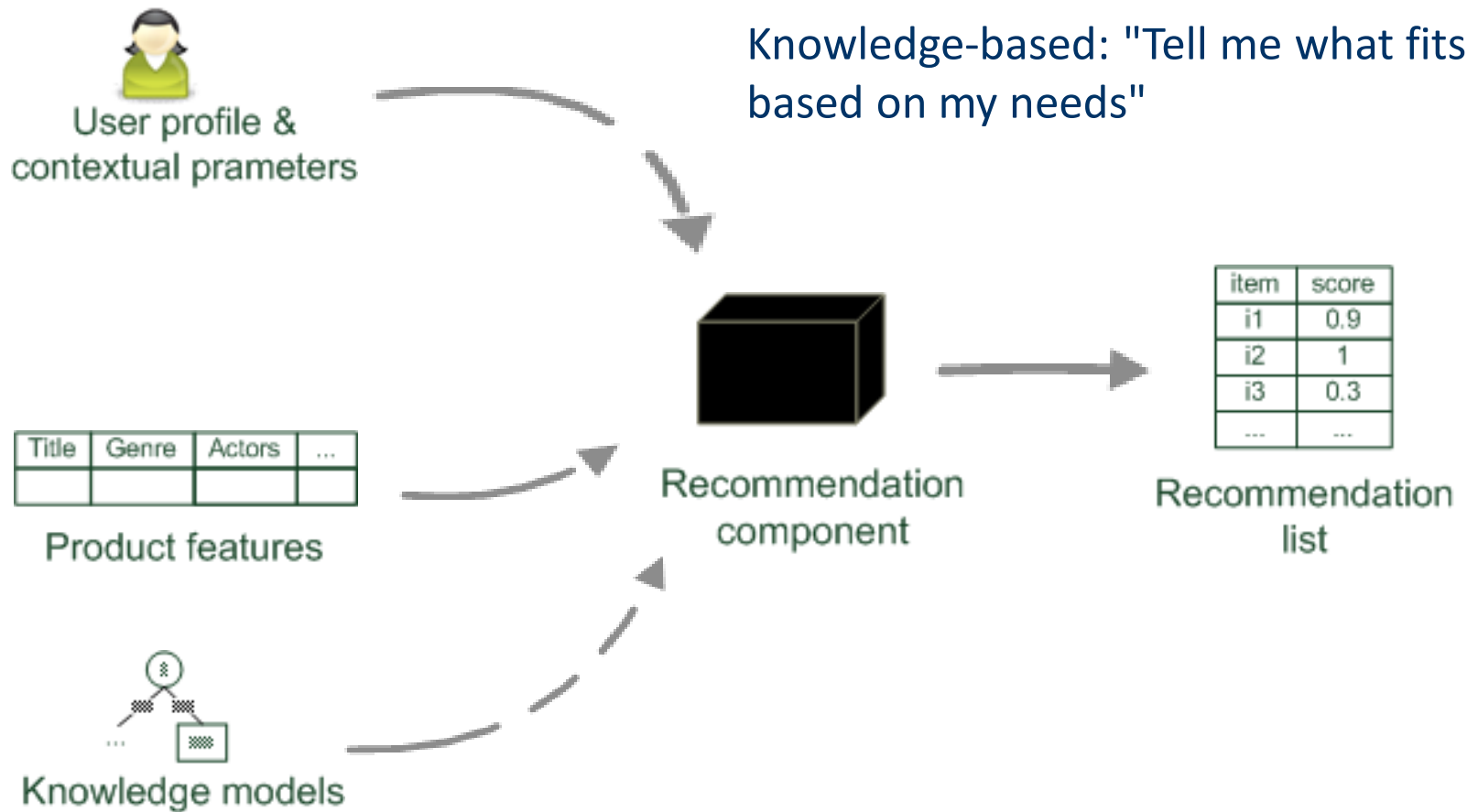


# Paradigms of recommender systems



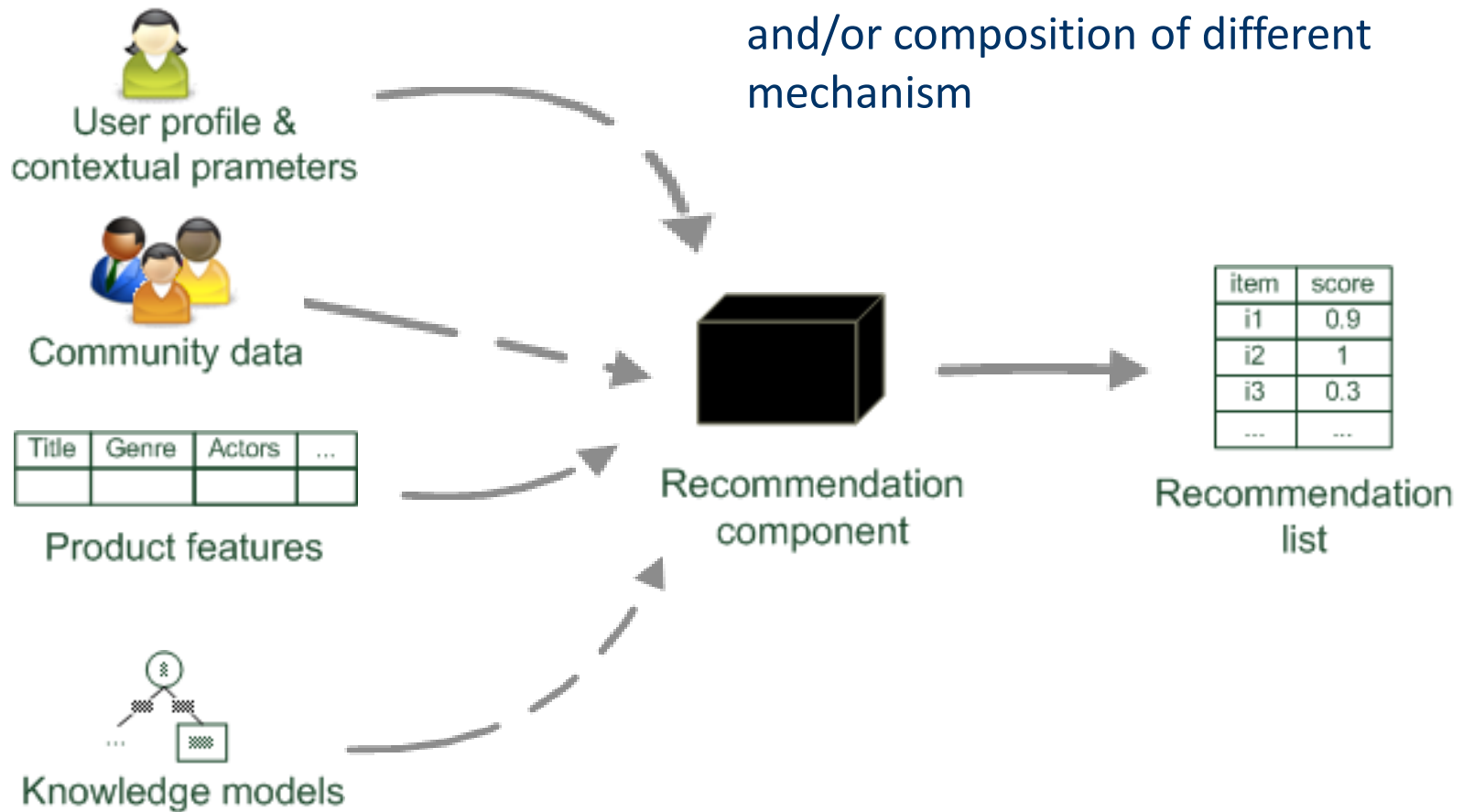


# Paradigms of recommender systems



# Paradigms of recommender systems

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



# Collaborative Filtering

Introduction to Recommender Systems

# 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:

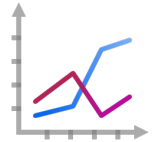
	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

- 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  $1$

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

# Measuring user similarity (2)

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

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

sim = 0.85

$$\bar{r}_a = \frac{5+3+4+4}{4} = 4, \quad \bar{r}_b = \frac{3+1+2+3}{4} = 2.25$$

$$\begin{aligned}
 \text{sim}(a, b) &= \frac{\sum_{p \in P} (r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P} (r_{a,p} - \bar{r}_a)^2} \sqrt{\sum_{p \in P} (r_{b,p} - \bar{r}_b)^2}} \\
 &= \frac{(5 - 4)(3 - 2.25) + (3 - 4)(1 - 2.25) + (4 - 4)(2 - 2.25) + (4 - 4)(3 - 2.25)}{\sqrt{(5 - 4)^2 + (3 - 4)^2 + (4 - 4)^2 + (4 - 4)^2} \sqrt{(3 - 2.25)^2 + (1 - 2.25)^2 + (2 - 2.25)^2 + (3 - 2.25)^2}} \\
 &= \frac{2}{\sqrt{2} \sqrt{2.75}} \\
 &= 0.85
 \end{aligned}$$

# Measuring user similarity (3)

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  $b$

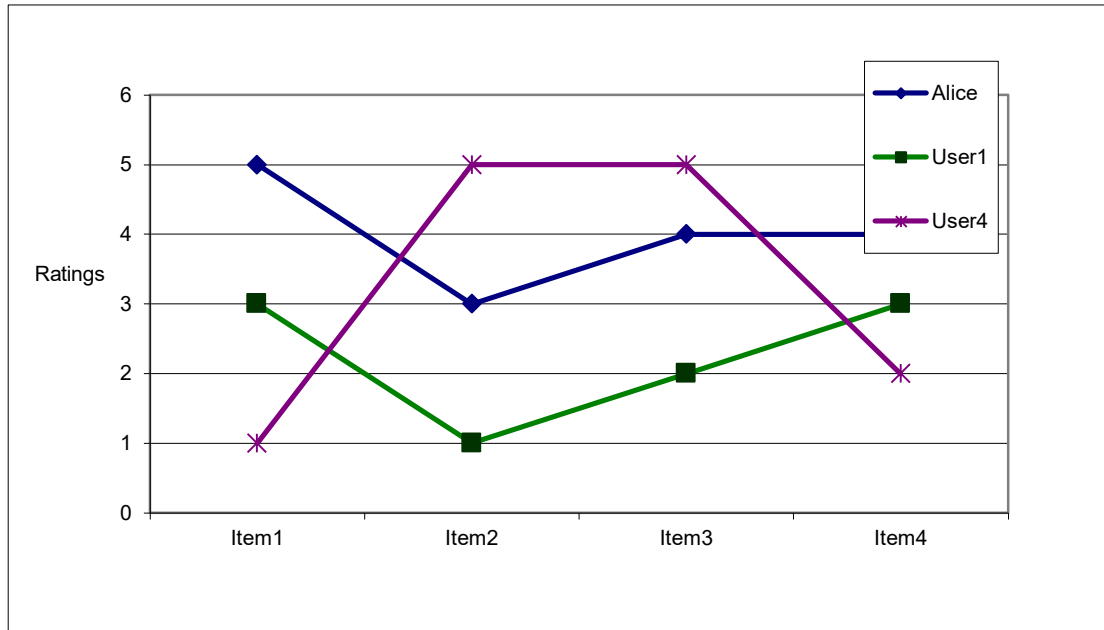
- Possible similarity values between  $-1$  and  $1$

	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



# 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:

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



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

# 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

# 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

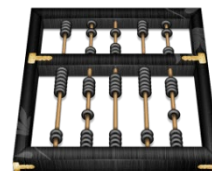
$$\text{sim}(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$



## Adjusted cosine similarity

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

$$\text{sim}(\vec{a}, \vec{b}) = \frac{\sum_{u \in U} (r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,a} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,b} - \bar{r}_u)^2}}$$





# Making predictions

A common prediction function:

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



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)

# 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
  - Limit the neighborhood size (might affect recommendation accuracy)

# More on ratings – 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?

# More on ratings – 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

For example, the Netflix Prize rating data in the user/movie rating matrix, you get  $500,000 * 17000 = 8500\text{M}$  positions Out of which only 100M are not 0's

## 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

# Model-based CF

- 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

# 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?

# Latent Factor Model-- Background

## MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS

Yehuda Koren, *Yahoo Research*  
Robert Bell and Chris Volinsky, *AT&T Labs—Research*

As the Netflix Prize competition has demonstrated, matrix factorization models are superior to classic nearest-neighbor techniques for producing product recommendations, allowing the incorporation of additional information such as implicit feedback, temporal effects, and confidence levels.

**M**odern consumers are inundated with choices. Electronic retailers and content providers offer a huge selection of products, with unprecedented opportunities to meet a variety of special needs and tastes. Matching consumers with the most appropriate products is key to enhancing user satisfaction and loyalty. Therefore, more retailers have become interested in recommender systems, which analyze patterns of user interest in products to provide personalized recommendations that suit a user's taste. Because good personalized recommendations can add another dimension to the user experience, e-commerce leaders like Amazon.com and Netflix have made recommender systems a salient part of their websites.

Such systems are particularly useful for entertainment products such as movies, music, and TV shows. Many customers will view the same movie, and each customer is likely to view numerous different movies. Customers have proven willing to indicate their level of satisfaction with particular movies, so a huge volume of data is available about which movies appeal to which customers. Companies can analyze this data to recommend movies to particular customers.

### RECOMMENDER SYSTEM STRATEGIES

Broadly speaking, recommender systems are based on one of two strategies. The *content filtering* approach creates a profile for each user or product to characterize its nature. For example, a movie profile could include attributes regarding its genre, the participating actors, its box office popularity, and so forth. User profiles might include demographic information or answers provided on a suitable questionnaire. The profiles allow programs to associate users with matching products. Of course, content-based strategies require gathering external information that might not be available or easy to collect.

A known successful realization of content filtering is the Music Genome Project, which is used for the Internet radio service Pandora.com. A trained music analyst scores

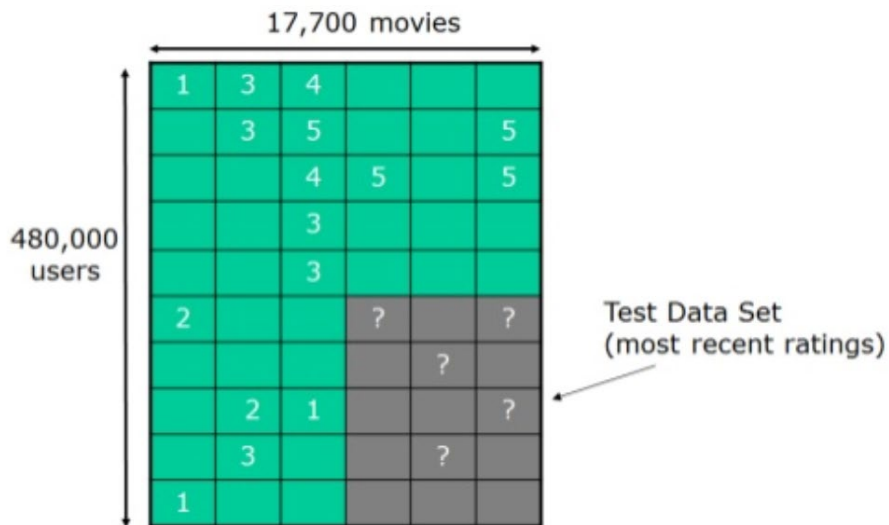
Yehuda Koren

Yahoo Research Robert Bell and Chris Volinsky, AT&T Labs-Research Paper published in August 2009 Authors won the grand Netflix Prize in September 2009



# LFM-based CF

- Netflix Prize (\$1 Million Contest), 2006-2009



- Find features that describe the characteristics of rated objects
- Item characteristics and user preferences are described with numerical factor values
- Assumption: Ratings can be inferred from a model put together from a smaller number of parameters

# LFM-based CF

- Items and users are associated with a factor vector
- Dot product captures the user's estimated interest in the item:

$$\hat{r}_{ui} = q_i^T p_u$$

- Challenge: How to compute a mapping of items and users to factor vectors?
- Approaches:
  - Singular Value Decomposition (SVD)
  - Matrix Factorization

# LFM: Dimensionality Reduction

- 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  $100$ )
- Application of Dimensionality Reduction in Recommender Systems (B. Sarwar et al., WebKDD Workshop, 2000)

# Matrix factorization

Informally, the SVD theorem (Golub and Kahan 1965) states that a given matrix  $M$  can be decomposed into a product of three matrices as follows

$$M = U \times \Sigma \times V^T$$

where  $U$  and  $V$  are called *left* and *right singular vectors* and the values of the diagonal of  $\Sigma$  are called the *singular values*

We can approximate the full matrix by observing only the most important features – those with the largest singular values

In the example, we calculate  $U$ ,  $V$ , and  $\Sigma$  (with the help of some linear algebra software) but retain only the two most important features by taking only the first two columns of  $U$  and  $V^T$

# Example for SVD-based recommendation

- SVD:  $M_k = U_k \times \Sigma_k \times V_k^T$

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

$V_k^T$	Terminator	Die Hard	Twins	Eat Pray Love	Pretty Woman
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

- Prediction:  $\hat{r}_{ui} = \bar{r}_u + U_k(\text{Alice}) \times \Sigma_k \times V_k^T(\text{EPL})$   
 $= 3 + 0.84 = 3.84$

$\Sigma_k$	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

# The projection of $U$ and $V^T$ in the 2 dimensional space $(U_2, V_2^T)$



# SVD Problems

Conventional SVD is undefined for incomplete matrices  
→ overfitting

Imputation to fill in missing values

- Increases the amount of data → increases computational cost

We need an approach that can simply ignore missing ratings

# Matrix Factorization

$$\min_{q^*, p^*} \sum_{(u,i) \in K} (r_{ui} - q_i^T p_u)^2$$



$$\min_{q^*, p^*} \sum_{(u,i) \in K} (r_{ui} - q_i^T p_u)^2 + \lambda (\|q_i\|^2 + \|p_u\|^2)$$

known rating of user  $u$  for item  $i$ :  $r_{ui}$

predicted rating:  $\hat{r}_{ui} = q_i^T p_u$

constant to control the extend of regularization, which is determined by cross-validation:

$\lambda$



# Matrix Factorization

## Learning Strategies

### Stochastic gradient descent

- Modification of parameters ( $q_i$ ,  $p_u$ ) relative to prediction error
- Recommended algorithm

### Alternating least squares

- Allows massive parallelization
- Better for densely filled matrices

# Matrix Factorization

## Calculation of the prediction error

- Error = actual rating – predicted rating

$$e_{ui} = r_{ui} - q_i^T p_u$$

## Modification

- By magnitude proportional to  $\gamma$
- In the opposite direction of the gradient

$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$$

# Matrix Factorization

Handling bias

Item or user specific rating variations are called biases

Example:

- Alice rates no movie with more than 2 (out of 5)
- Movie X is hyped and rated with 5 only

Matrix factorization allows modeling of biases

Including bias parameters in the prediction:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

# Discussion about dimensionality reduction (Sarwar et al. 2000a)

## Matrix factorization

- Generate low-rank approximation of matrix
- Detection of latent factors
- Projecting items and users in the same n-dimensional space

Prediction quality can decrease because...

- the original ratings are not taken into account

Prediction quality can increase as a consequence of...

- filtering out some "noise" in the data and
- detecting nontrivial correlations in the data

Depends on the right choice of the amount of data reduction

- number of singular values in the SVD approach
- Parameters can be determined and fine-tuned only based on experiments in a certain domain
- Koren et al. 2009 talk about 20 to 100 factors that are derived from the rating patterns

# Association rule mining

Commonly used for shopping behavior analysis

- aims at detection of rules such as

*"If a customer purchases beer then he also buys diapers  
in 70% of the cases"*

## Association rule mining algorithms

- can detect rules of the form  $X \rightarrow Y$  (e.g., beer  $\rightarrow$  diapers) from a set of sales transactions  $D = \{t_1, t_2, \dots, t_n\}$
- measure of quality: support, confidence
  - used e.g. as a threshold to cut off unimportant rules
- let  $\sigma(X) = \frac{|\{x | x \subseteq t_i, t_i \in D\}|}{|D|}$
- support =  $\frac{\sigma(X \cup Y)}{|D|}$ , confidence =  $\frac{\sigma(X \cup Y)}{\sigma(X)}$

# Recommendation based on Association Rule Mining

## Simplest approach

- transform 5-point ratings into binary ratings (1 = above user average)

## Mine rules such as

- Item1  $\rightarrow$  Item5
  - support (2/4), confidence (2/2) (without Alice)

	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

## 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  $i$
- 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  $P(\text{Item5}=1 \mid X)$ , where
  - $X$  = Alice's previous ratings = (Item1 =1, Item2=3, Item3= ... )
- Can be estimated based on Bayes' Theorem

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



- Assumption: Ratings are independent (?)

# 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

$X = (\text{Item1} = 1, \text{Item2} = 3, \text{Item3} = \dots)$

$$\begin{aligned}
 &P(X|\text{Item5} = 1) \\
 &= P(\text{Item1} = 1|\text{Item5} = 1) \times P(\text{Item2} = 3|\text{Item5} = 1) \\
 &\times P(\text{Item3} = 3|\text{Item5} = 1) \times P(\text{Item4} = 2|\text{Item5} = 1) \\
 &= \frac{2}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \approx 0.125
 \end{aligned}$$



$$\begin{aligned}
 &P(X|\text{Item5} = 2) \\
 &= P(\text{Item1} = 1|\text{Item5} = 2) \times P(\text{Item2} = 3|\text{Item5} = 2) \\
 &\times P(\text{Item3} = 3|\text{Item5} = 2) \times P(\text{Item4} = 2|\text{Item5} = 2) \\
 &= \frac{0}{0} \times \dots \times \dots \times \dots = 0
 \end{aligned}$$

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



# The Google News personalization engine

The screenshot displays the Google News interface. On the left is a sidebar with navigation options: 'Top stories', 'For you', 'Favourites', 'Saved searches', and a list of categories (Australia, World, Local, Business, Technology, Entertainment, Sports, Science, Health). The main content area is titled 'Headlines' and features three news stories. The first story is about an Ipsos poll regarding the Morrison government. The second story is about a homicide investigation into the death of a woman. The third story is about a woman saving her children from a car that caught fire. To the right of the headlines is a weather widget for Sydney, showing a sunny forecast with a temperature of 23°C and a 5-day outlook. Below the weather widget is a 'In the news' section with a grid of topic tags such as 'Gold Coast Football Club', 'North Melbourne Football Club', and 'European Union'.

**Google News**

Search for topics, locations & sources

**Headlines**

**Ipsos poll: 53-47 result puts Morrison government on course for major election defeat**  
Sydney Morning Herald • 1 hour ago

- Labor says Government delaying election call so it can use taxpayer funds to pay for advertising policies  
ABC News • 3 hours ago
- Australian federal election date: Scott Morrison stalls on election call  
NEWS.com.au • 10 hours ago
- Parliament has been enslaved by its fetishes – and it's time to end the downward spiral  
The Guardian • Yesterday • Opinion
- 'Shorten wants to end the weekend': Morrison attacks Labor's electric vehicle policy  
The Guardian • 3 hours ago

[View full coverage](#)

**'Terrifying ordeal': homicide detectives probe death of Sydenham woman Vicki Ramadan**  
Sydney Morning Herald • 5 hours ago

- Dead Melbourne woman identified as 77-year-old Vicki Ramadan  
9news.com.au • 7 hours ago

[View full coverage](#)

**'Superwoman' Canberra mum drags kids from car seconds before it explodes into flames**  
9news.com.au • 8 hours ago

- Terrifying moment mum pulls kids out of car seconds before it bursts into flames  
Yahoo News Australia • 7 hours ago

[View full coverage](#)

**Sydney**

Sunny  
23°C

Today	Mon	Tue	Wed	Thu
26°C 18°C	30°C 19°C	24°C 15°C	19°C 15°C	21°C 15°C

C | F | K [More on weather.com](#)

**In the news**

Gold Coast Football Club  
North Melbourne Football Club  
European Union  
Melbourne Storm  
Scott Morrison  
Canterbury-Bankstown Bulldogs  
National Rugby League  
Israel  
Western Bulldogs  
Brexit

# Google News portal (1)

Aggregates news articles from several thousand sources

Displays them to signed-in users in a personalized way

Collaborative recommendation approach based on

- the click history of the active user and
- the history of the larger community

## Main challenges

- Vast number of articles and users
- Generate recommendation list in real time (at most one second)
- Constant stream of new items
- Immediately react to user interaction

Significant efforts with respect to algorithms, engineering, and parallelization are required

# Google News portal (2)

Pure memory-based approaches are not directly applicable and for model-based approaches, the problem of continuous model updates must be solved

A combination of model- and memory-based techniques is used

Model-based part: Two clustering techniques are used

- Probabilistic Latent Semantic Indexing (PLSI) as proposed by (Hofmann 2004)
- MinHash as a hashing method

Memory-based part: Analyze story *co-visits* for dealing with new users

Google's MapReduce technique is used for parallelization in order to make computation scalable

# Limitations of CF

- Cold Start: have enough other users already in the system to find a match. New items need to get enough ratings.
- Popularity Bias: hard to recommend items to someone with unique tastes
- Trends to recommend popular items (items from the tail don't get so much data)

# Cold Start

- New User Problem: To make accurate recommendations, the system must first learn the user's preferences from the ratings.
- Several techniques proposed to address this. Most use the hybrid recommendation approach, which combines content-based and collaborative techniques
- New item problem: New items are added regularly to recommender systems. Until new item is rated by a substantial number of users, the recommender system is not able to recommend it.

# 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*-based CF is an example for model-based approaches

# Content-based recommendation

Introduction to Recommender Systems

# Content-based recommendation

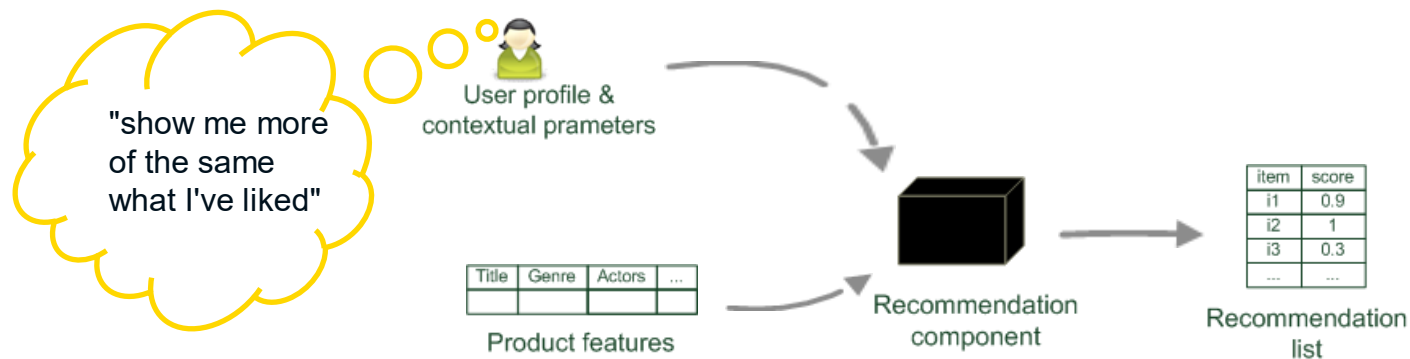
While CF – methods do not require any information about the items,  
– it might be reasonable to exploit such information; and  
– recommend fantasy novels to people who liked fantasy novels in the past

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)

The task:

- learn user preferences
- locate/recommend items that are "similar" to the user preferences





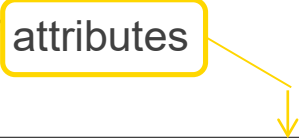
# What is the "content"?

Most CB-recommendation techniques were applied to recommending text documents.

- Like web pages or newsgroup messages for example.

Content of items can also be represented as text documents.

- With textual descriptions of their basic characteristics.
- Structured: Each item is described by the same set of attributes



Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

- Unstructured: free-text description.

# Content representation and item similarities

## Item representation

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

## User profile

Title	Genre	Author	Type	Price	Keywords
...	Fiction	Brunonia, Barry, Ken Follett	Paperback	25.65	Detective, murder, New York

$keywords(b_j)$   
describes Book  $b_j$   
with a set of  
keywords



## Simple approach

- Compute the similarity of an unseen item with the user profile based on the keyword overlap (e.g. using the Dice coefficient)
- Or use and combine multiple metrics



$$\frac{2 \times |keywords(b_i) \cap keywords(b_j)|}{|keywords(b_i)| + |keywords(b_j)|}$$

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

Simple keyword representation has its problems

- in particular when automatically extracted as
  - not every word has similar importance
  - longer documents have a higher chance to have an overlap with the user profile

Standard measure: TF-IDF

- Encodes text documents in multi-dimensional Euclidian space
  - weighted term vector
- TF: Measures, how often a term appears (density in a document)
  - assuming that important terms appear more often
  - normalization has to be done in order to take document length into account
- IDF: Aims to reduce the weight of terms that appear in all documents

# TF-IDF II

Given a keyword  $i$  and a document  $j$

$$TF(i, j)$$

–term frequency of keyword  $i$  in document  $j$

$$IDF(i)$$

–inverse document frequency calculated as  $IDF(i) = \log \frac{N}{n(i)}$

»  $N$  : number of all recommendable documents

»  $n(i)$  : number of documents from  $N$  in which keyword  $i$  appears

$$TF - IDF$$

• is calculated as:  $TF-IDF(i, j) = TF(i, j) * IDF(i)$

# Example TF-IDF representation

Term frequency:

- Each document is a **count vector** in  $\mathbb{N}^{|v|}$

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	1.51	0	3	5	5	1
worser	1.37	0	1	1	1	0

Vector  $v$  with dimension  $|v| = 7$

# Example TF-IDF representation

## Combined TF-IDF weights

- Each document is now represented by a real-valued vector of *TF-IDF* weights  $\in \mathbb{R}^{|v|}$

		Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	
Antony	157	73	0	0	0	0		
Brutus	4		Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Caesar	232							
Calpurnia	0							
Cleopatra	57							
mercy	1.5	Antony	5.25	3.18	0	0	0	0.35
worser	1.3	Brutus	1.21	6.1	0	1	0	0
		Caesar	8.59	2.54	0	1.51	0.25	0
		Calpurnia	0	1.54	0	0	0	0
		Cleopatra	2.85	0	0	0	0	0
		mercy	1.51	0	1.9	0.12	5.25	0.88
		worser	1.37	0	0.11	4.15	0.25	1.95

# Improving the vector space model

Vectors are usually long and sparse

remove stop words

- They will appear in nearly all documents.
- e.g. "a", "the", "on", ...

use stemming

- Aims to replace variants of words by their common stem
- e.g. "went" → "go", "stemming" → "stem", ...

size cut-offs

- only use top n most representative words to remove "noise" from data
- e.g. use top 100 words

# Improving the vector space model II

Use lexical knowledge, use more elaborate methods for feature selection

- Remove words that are not relevant in the domain

Detection of phrases as terms

- More descriptive for a text than single words
- e.g. "United Nations"

Limitations

- semantic meaning remains unknown
- example: usage of a word in a negative context
  - "there is nothing on the menu that a vegetarian would like.."
  - The word "vegetarian" will receive a higher weight then desired
    - ➡ an unintended match with a user interested in vegetarian restaurants



# Cosine similarity

Usual similarity metric to compare vectors: Cosine similarity (angle)

- Cosine similarity is calculated based on the angle between the vectors

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

Adjusted cosine similarity

- take average user ratings into account ( $\bar{r}_u$ ), transform the original ratings
- U: set of users who have rated both items a and b

$$\text{sim}(\vec{a}, \vec{b}) = \frac{\sum_{u \in U} (r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,a} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,b} - \bar{r}_u)^2}}$$

# Recommending items

## Simple method: nearest neighbors

- Given a set of documents  $D$  already rated by the user (like/dislike)
  - Either explicitly via user interface
  - Or implicitly by monitoring user's behavior
- Find the  $n$  nearest neighbors of an not-yet-seen item  $i$  in  $D$ 
  - Use similarity measures (like cosine similarity) to capture similarity of two documents
- Take these neighbors to predict a rating for  $i$ 
  - e.g.  $k = 5$  most similar items to  $i$ .  
4 of  $k$  items were liked by current user ➡ item  $i$  will also be liked by this user
- Variations:
  - Varying neighborhood size  $k$
  - lower/upper similarity thresholds to prevent system from recommending items the user already has seen
- Good to model short-term interests / follow-up stories
- Used in combination with method to model long-term preferences

# Limitations of content-based recommendation methods

Keywords alone may not be sufficient to judge quality/relevance of a document or web page

- up-to-date-ness, usability, aesthetics, writing style
- content may also be limited / too short
- content may not be automatically extractable (multimedia)

Ramp-up phase required

- Some training data is still required
- Web 2.0: Use other sources to learn the user preferences

Overspecialization

- Algorithms tend to propose "more of the same"
- Or: too similar news items

# Probabilistic Model (Basic)

	Item1	Item2	Item3	Item5	Item5
Alice	5	3	2	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

- $X = (\text{Item1} = 5, \text{Item2} = 3, \text{Item3} = \dots)$
- $P(X | \text{Item5} = 1) =$   
 $P(\text{Item1} = 5 | \text{item5} = 1) * P(\text{Item2} = 3 | \text{item5} = 1) * P(\text{item3} = 2 | \text{item5} = 1) * P(\text{item4} = 4 | \text{item5} = 1)$
- $P(X | \text{item5} = 2) = \dots \dots$
- $\dots$
- $P(X | \text{item5} = 5) = \dots \dots$

# Classifier

- Treat recommendation as a classification problem (rating prediction)
- Classifiers are general computational models trained using positive and negative examples
- They may take in inputs:
  - Vector of item features (action/adventure, Bruce Willis)
  - Preferences of customers (like action/adventure)
  - Relations among item e.g., logistic regression, Bayesian networks, SVM...

# Classifier

Classifiers can be used in CF and CB recommendations

- Pros:

- Versatile: can be combined with other methods to improve accuracy of recommendations

- Cons:

- Need a relevant training set
- May overfit (regularization)

# Pros

- 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 and unpopular items
  - No first-rater problem
- Can provide explanations of recommended items by listing content-features that caused an item to be recommended.

# Cons

- Requires content that can be encoded as meaningful features
- Some kind of item are not amenable to easy feature extraction methods (e.g., movies, music)
- Even for texts, IR techniques can't consider multimedia information, aesthetic qualities, download time
  - If you rate positively a page it could be not related to the presence of certain keywords
- Users' tastes must be represented as learnable function of these content features
- Hard to exploit quality of judgements of other users
- Difficult to implement serendipity
- Easy to overfit (e.g. a user with few data points we may pigeon hole her)