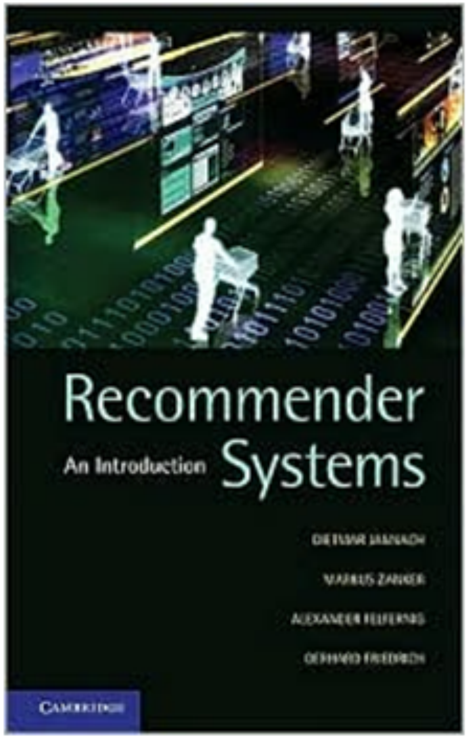
RECOMMENDER SYSTEMS

CS 584: Recommender Systems
Fall 2016
Huzefa Rangwala, Ph.D.
http://www.cs.gmu.edu/~hrangwal

Slides Adapted/Borrowed from: Koren, Bell, Leskovec and **Dietmar Jannach**

Outline

- What are Recommender Systems ?
- Collaborative Filtering vs Content Based
- Hybrid/Knowledge Based
- Advanced Topics



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

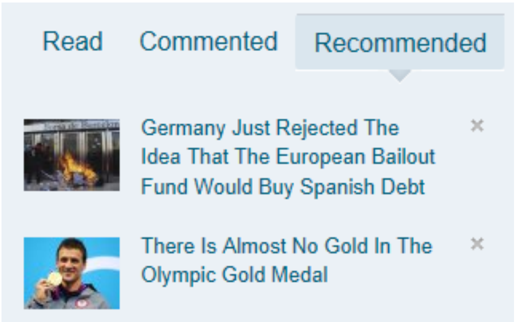
See All Prices

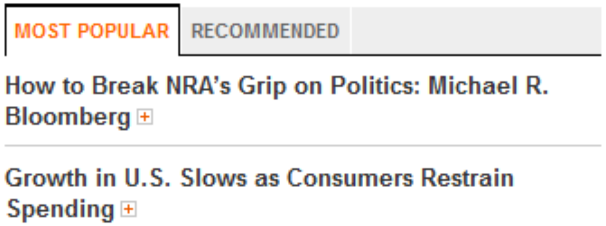
Table of Contents

Customers who bought this also bought

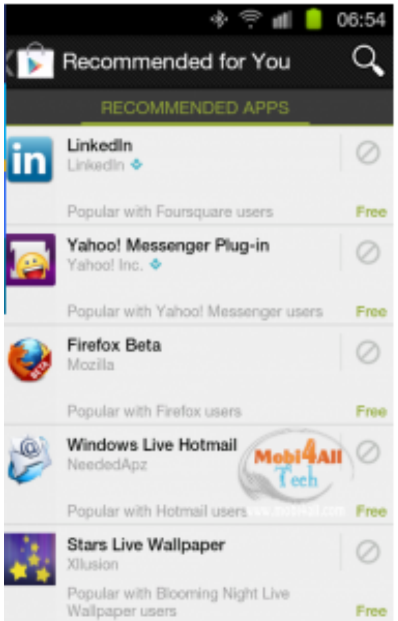
Recommender Systems

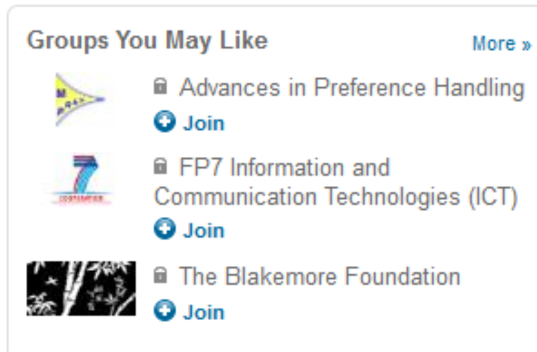
- Application areas

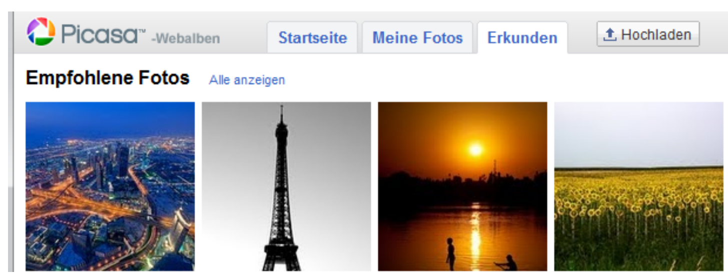




In the Social Web

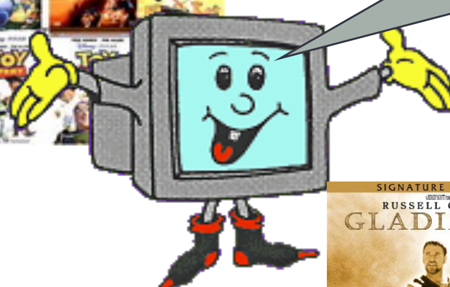


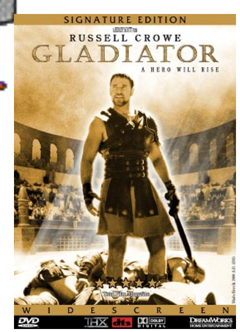




Movie RS

We Know What You Ought To Be Watching This Summer






Netflix Prize
Home
Rules
Leaderboard
Register
Update
Submit
Download
NETFLIX
Browse
Recommendations
Friends
Queue
Buy DVDs
Home
Genres
New Releases
Previews
Netflix Top 100
Movies For You
Randy, the following movies were chosen based on your interest in:
Bowling for Columbine
Carnivale: Season 1
Fahrenheit 9/11
The Big One
All Discs Guaranteed!
Welcome!
The Netflix Prize seeks to substantially improve the accuracy of predictions about how much someone is going to love a movie based on their movie preferences. Improve it enough and you win one (or more) Prizes. Winning the Netflix Prize improves our ability to connect people to the movies they love.
Read the Rules to see what is required to win the Prizes. If you are interested in joining the quest, you should register a team.
You should also read the frequently-asked questions about the Prize. And check out how various teams are doing on the Leaderboard.
Good luck and thanks for helping!
Add
Not Interested
Not Interested
Red Eye
Rear Window
Member Favorites
Easter Eggs
By Decade
By Studio
Movies You've Seen
Give a friend

Value of Recommender Systems [WPS]

- To the Customer ?
- To the Provider ?

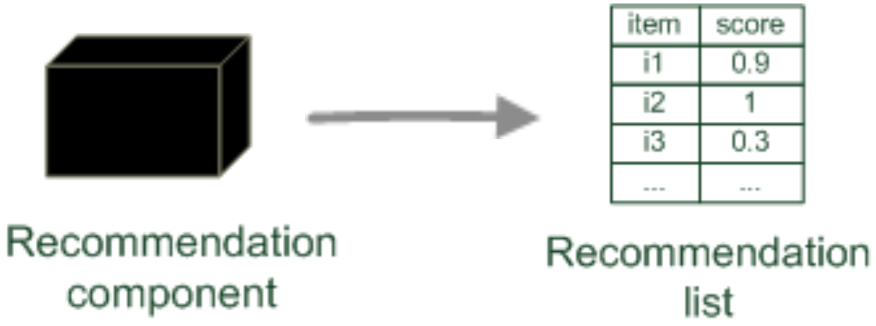
What information would you use to build one ? [WPS]

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)

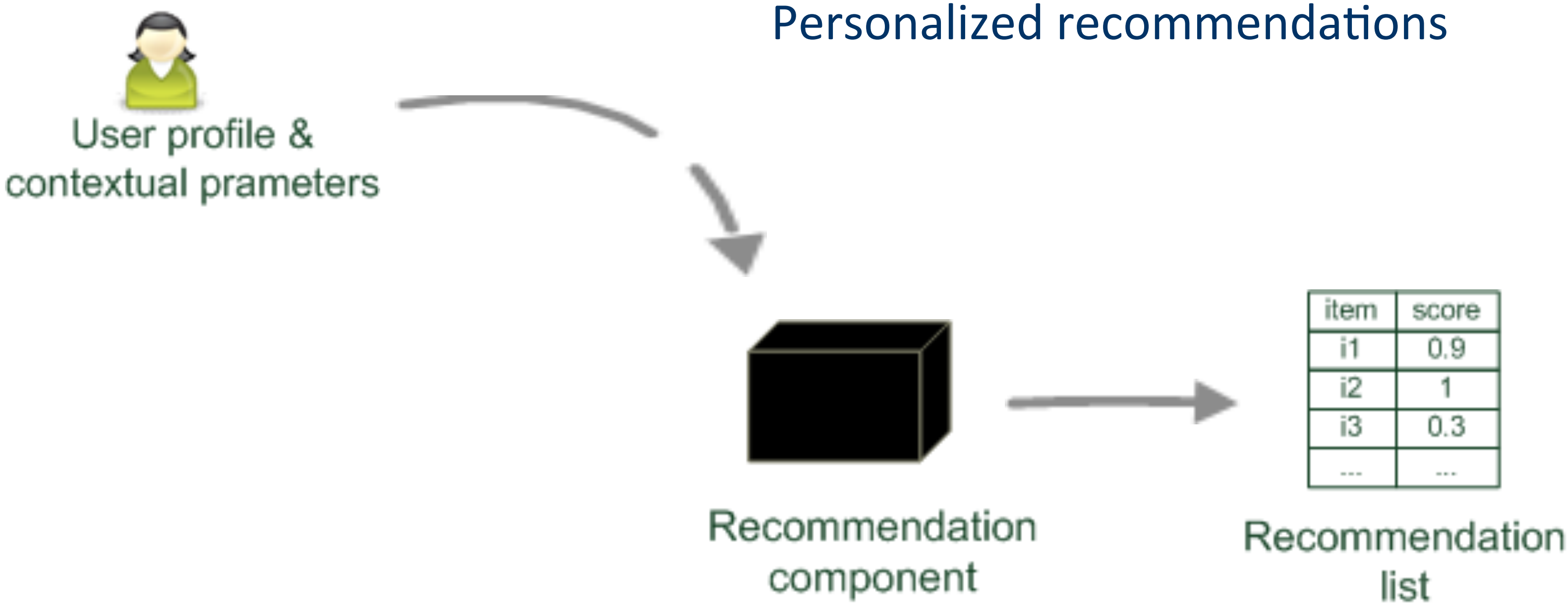
Paradigms of recommender systems

Recommender systems reduce information overload by estimating relevance



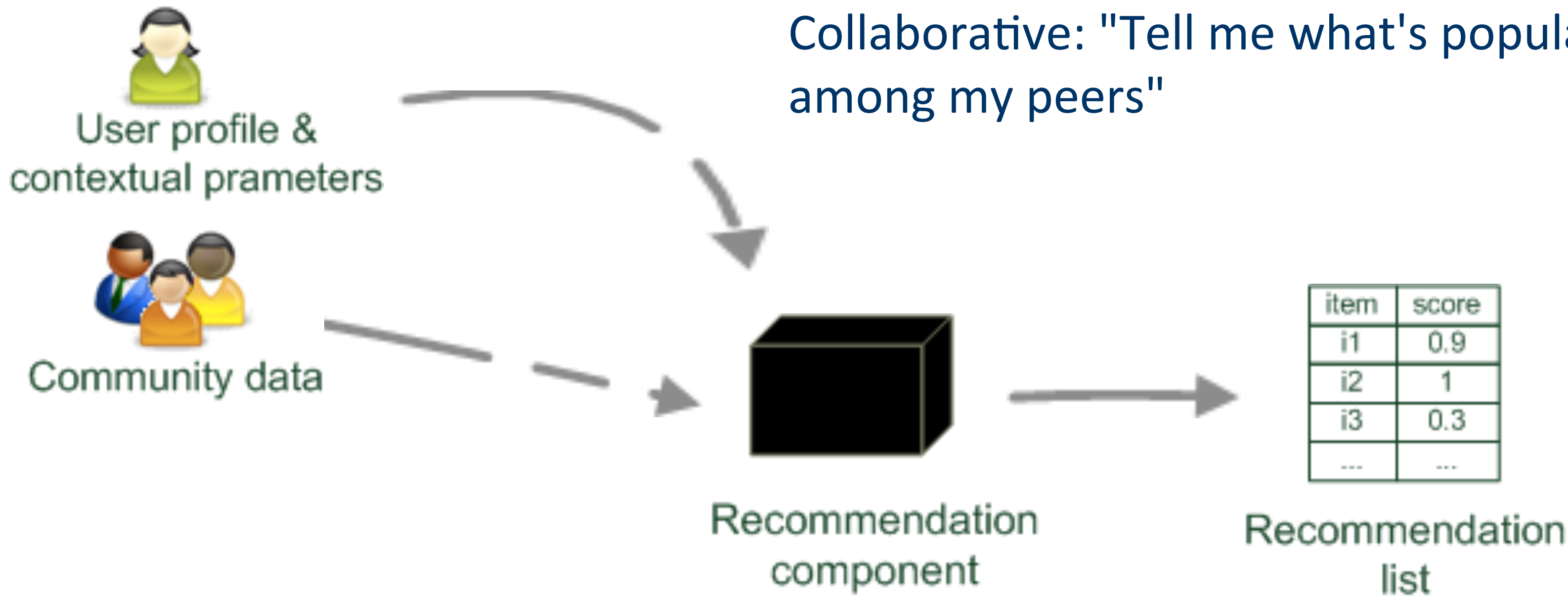
Paradigms of recommender systems

Personalized recommendations



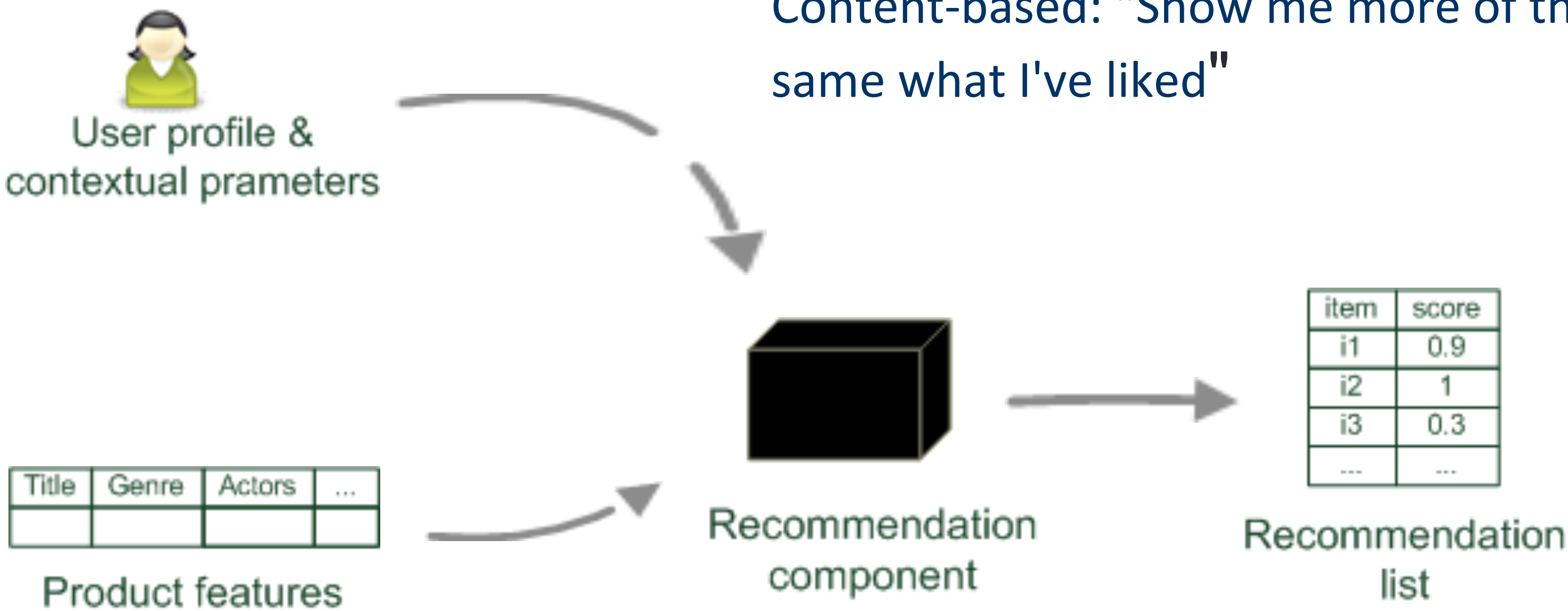
Paradigms of recommender systems

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



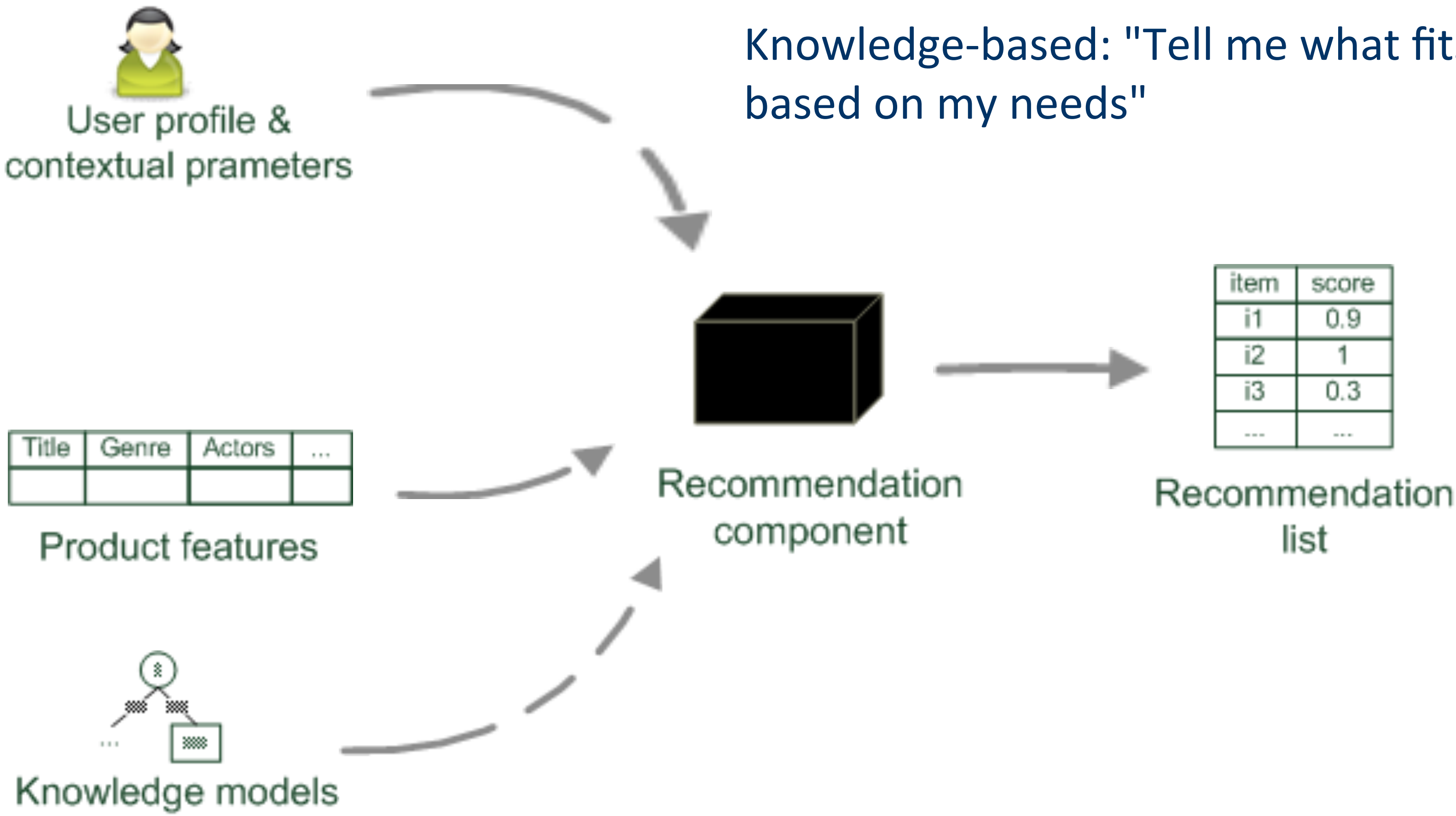
Paradigms of recommender systems

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



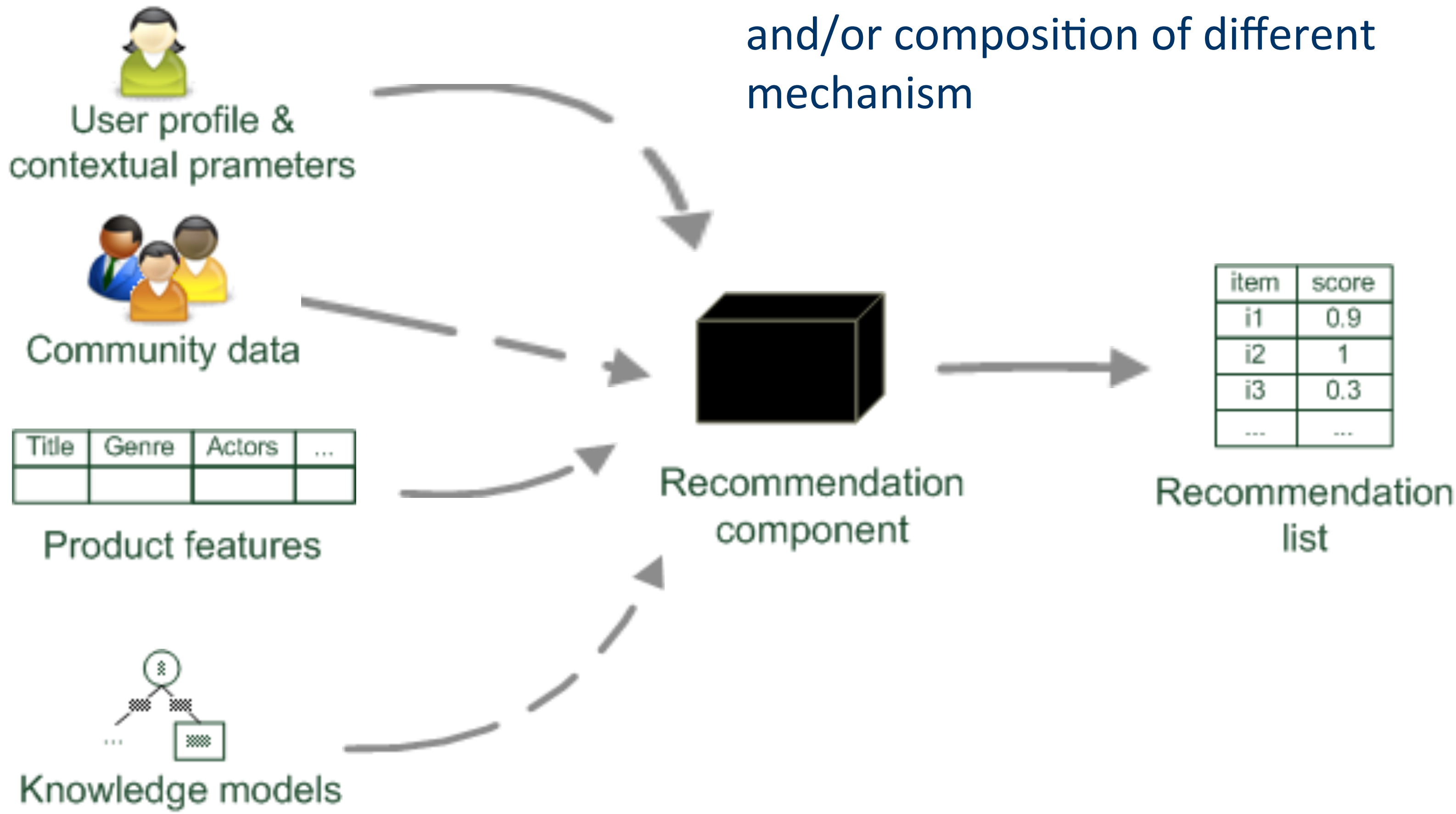
Paradigms of recommender systems

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



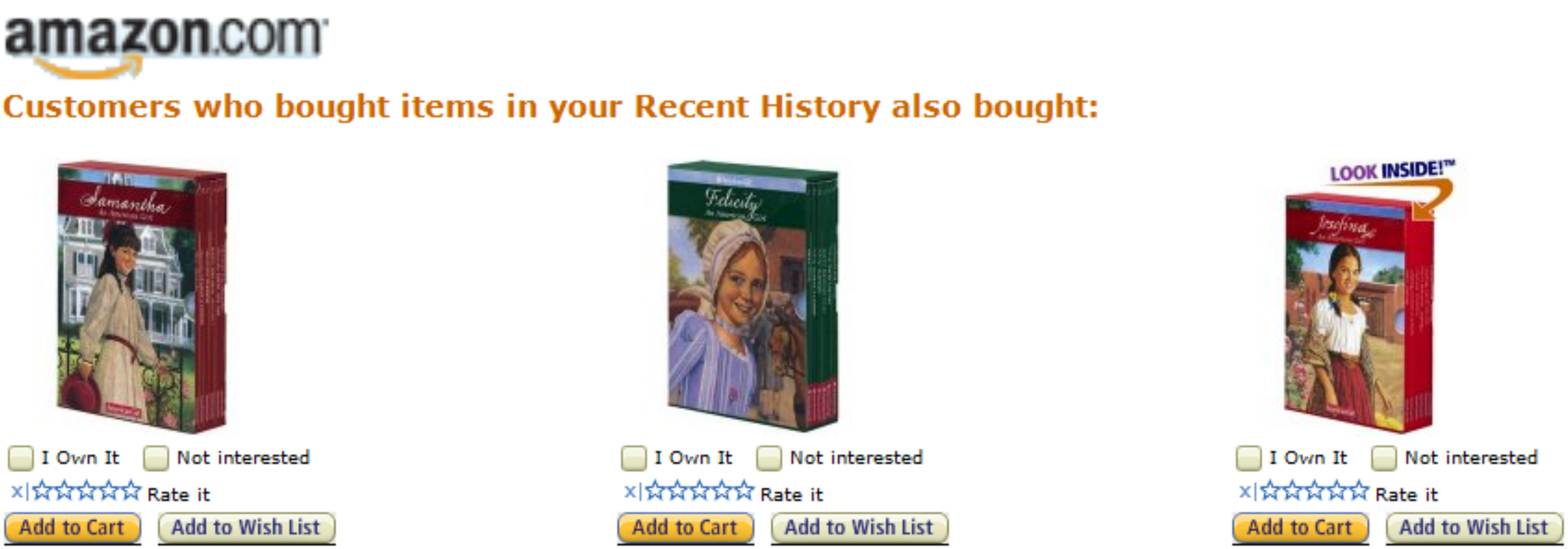
Paradigms of recommender systems

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



Collaborative filtering

- Recommend items based on past transactions of users
- Analyze relations between users and/or items
- Specific data characteristics are irrelevant
    - Domain-free: user/item attributes are not necessary
    - Can identify elusive aspects



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 nearest-neighbor collaborative filtering (1)

- 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 (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: **Pearson correlation**

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

a, b : users

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

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

Possible similarity values between -1 and 1; = user's

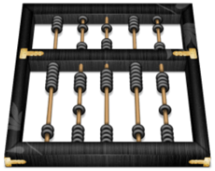
	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

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

Making predictions

- A common prediction function:

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



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

Making recommendations

- Making predictions is typically not the ultimate goal
- Usual approach (in academia)
  - Rank items based on their predicted ratings
- However
  - This might lead to the inclusion of (only) niche items
  - **In practice also:** Take item popularity into account
- Approaches
  - "Learning to rank"
    - Optimize according to a given rank evaluation metric (see later)

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

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 $O(m^2n)$
- 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

The cosine 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
- Similarity is calculated based on the angle between the vectors

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

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

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

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 *n* users)
    - Limit the size of the neighborhood (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 …
  - 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 to small to make good predictions
    - Assume "transitivity" of neighborhoods

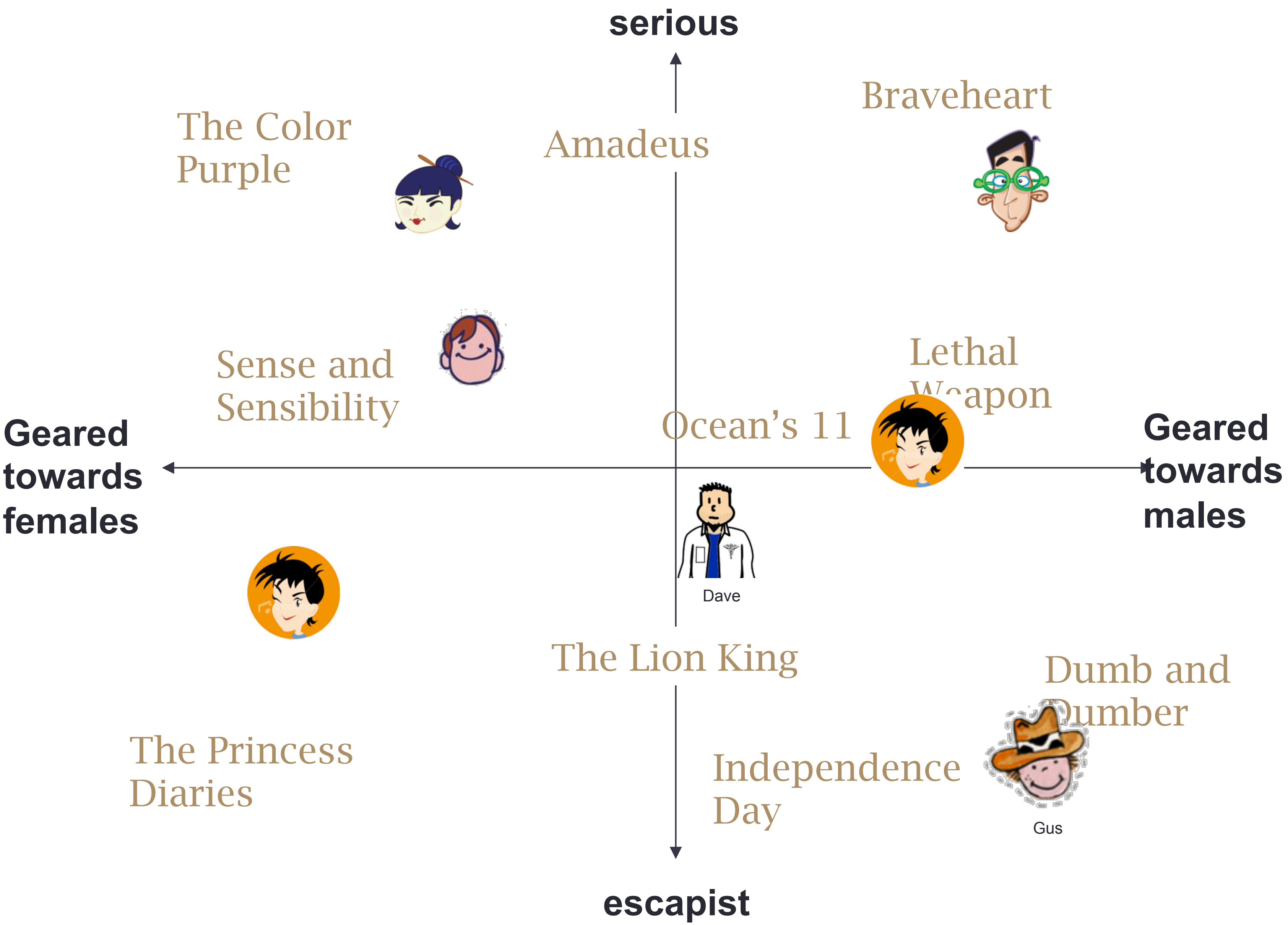
Model-based approaches

- 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?

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, …

Latent factor models



Latent factor models

users

items

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

~

~

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

•

users

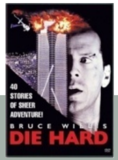
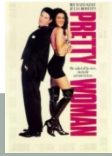
1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

A rank-3 SVD approximation

Matrix factorization

- 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					
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

Factorization meets the neighborhood: a multifaceted collaborative filtering model, Y. Koren, ACM SIGKDD

- **Stimulated by work on Netflix competition**
  - Prize of $1,000,000 for accuracy improvement of 10% RMSE compared to own Cinematch system
  - Very large dataset (~100M ratings, ~480K users , ~18K movies)
  - Last ratings/user withheld (set K)
- **Root mean squared error metric optimized to 0.8567**

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in K} (\hat{r}_{ui} - r_{ui})^2}{|K|}}$$

Factorization meets the neighborhood: a multifaceted collaborative filtering model, Y. Koren, ACM SIGKDD

- **Merges neighborhood models with latent factor models**
- **Latent factor models**
  - good to capture weak signals in the overall data
- **Neighborhood models**
  - good at detecting strong relationships between close items
- **Combination in one prediction single function**
  - Local search method such as stochastic gradient descent to determine parameters
  - Add penalty for high values to avoid over-fitting

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

$$\min_{p_*, q_*, b_*} \sum_{(u,i) \in K} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2)$$

Summarizing recent methods

- Recommendation is concerned with learning from noisy observations (x, y), where

$$f(x) = \hat{y}$$

  has to be determined such that $\sum_{\hat{y}} (\hat{y} - y)^2$ is minimal.

- A variety of different learning strategies have been applied trying to estimate f(x)
  - Non parametric neighborhood models
  - MF models, SVMs, Neural Networks, Bayesian Networks,…

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?

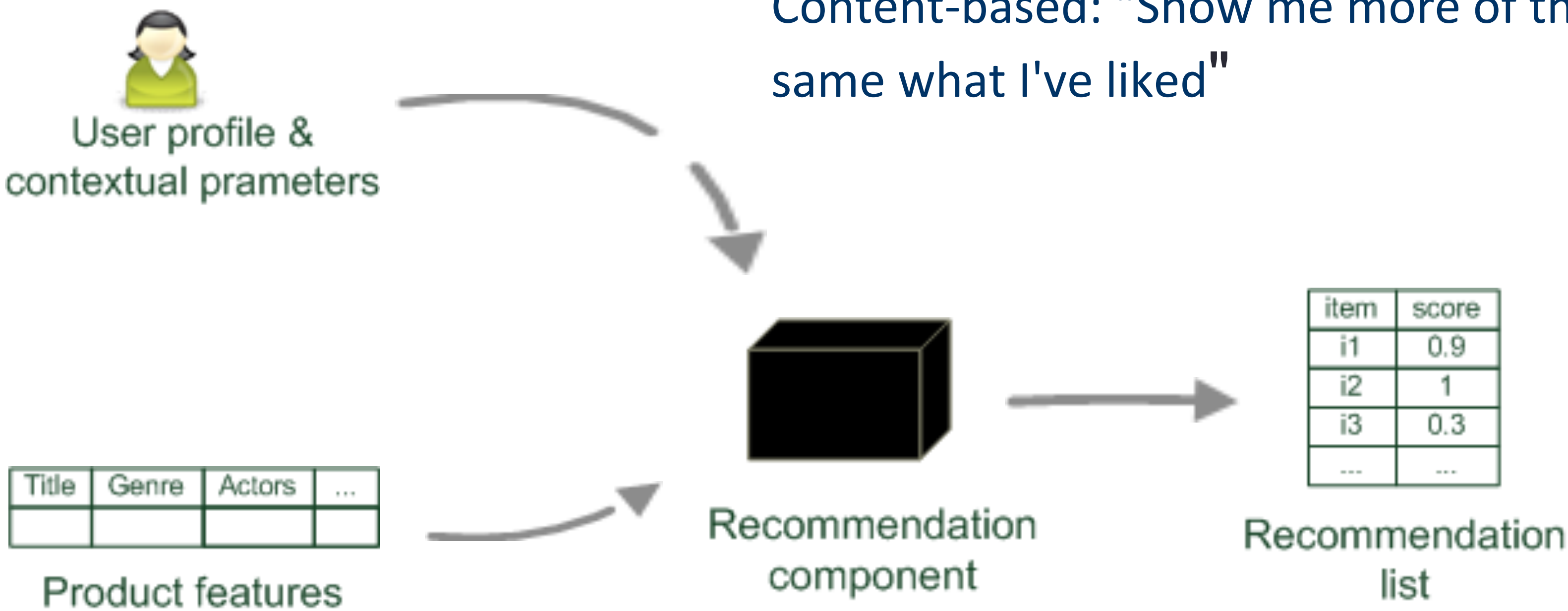
Content-based recommendation

Content-based recommendation

- Collaborative filtering does NOT require any information about the items,
  - However, it might be reasonable to exploit such information
  - E.g. 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

Paradigms of recommender systems

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



What is the "content"?

- The genre is actually not part of the content of a book
- Most CB-recommendation methods originate from Information Retrieval (IR) field:
  - The item descriptions are usually automatically extracted (important words)
  - Goal is to find and rank interesting text documents (news articles, web pages)

- Here:
  - Classical IR-based methods based on keywords
  - No expert recommendation knowledge involved
  - User profile (preferences) are rather learned than explicitly elicited

Content representation and item similarities

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
...					

Title	Genre	Author	Type	Price	Keywords
...	Fiction, Suspense	Brunonia Barry, Ken Follet, ..	Paperback	25.65	detective, murder, New York

- Simple approach
  - Compute the similarity of an unseen item with the user profile based on the keyword overlap (e.g. using the Dice coefficient)
  - $sim(b_i, b_j) = 2 * |keywords(bi) \cap keywords(bj)| / |keywords(bi)| + |keywords(bj)|$

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

- Simple keyword representation has its problems
  - In particular when automatically extracted because
    - 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 as 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

- 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*

- Inverse Document Frequency (IDF)
  - N: number of all recommendable documents
  - n(i): number of documents in which keyword *i* appears

Example TF-IDF representation

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
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

Figure taken from http://informationretrieval.org

Recommending items

- Simple method: nearest neighbors
  - Given a set of documents D already rated by the user (like/dislike)
    - Find the n nearest neighbors of a not-yet-seen item *i* in D
    - Take these ratings to predict a rating/vote for *i*
    - (Variations: neighborhood size, lower/upper similarity thresholds)
- Query-based retrieval: Rocchio's method
  - The SMART System: Users are allowed to rate (relevant/irrelevant) retrieved documents (feedback)
  - The system then learns a prototype of relevant/irrelevant documents
  - Queries are then automatically extended with additional terms/ weight of relevant documents

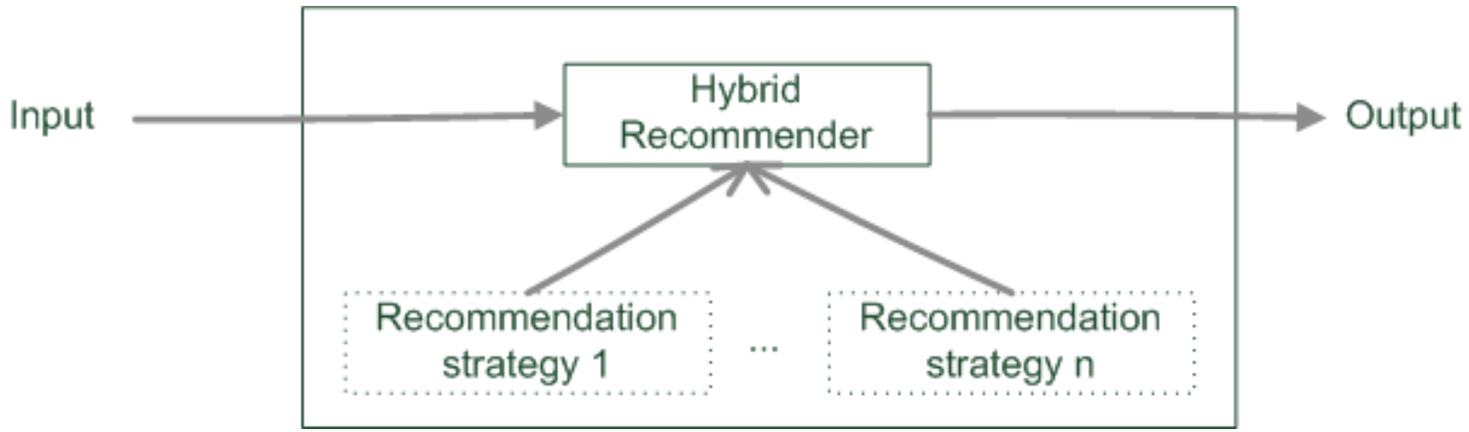
Limitations of content-based recommendation methods

- Keywords alone may not be sufficient to judge quality/relevance of a document or web page
  - Up-to-dateness, 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"
  - E.g. too similar news items

Hybridization Strategies

Monolithic hybridization design

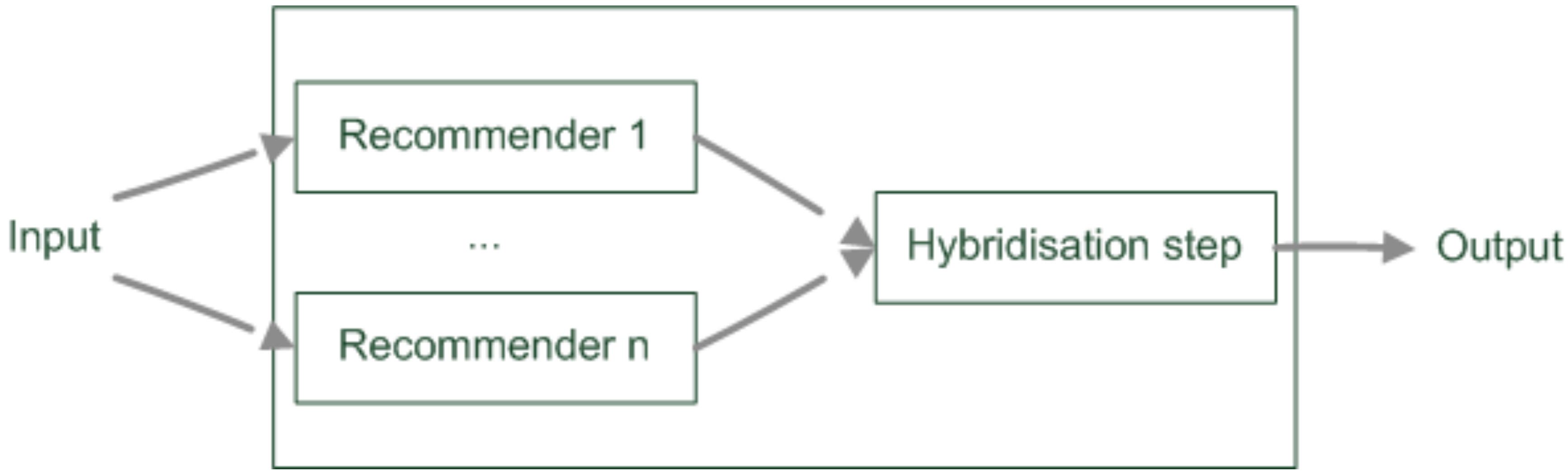
- Only a single recommendation component



- Hybridization is "virtual" in the sense that
  - Features/knowledge sources of different paradigms are combined

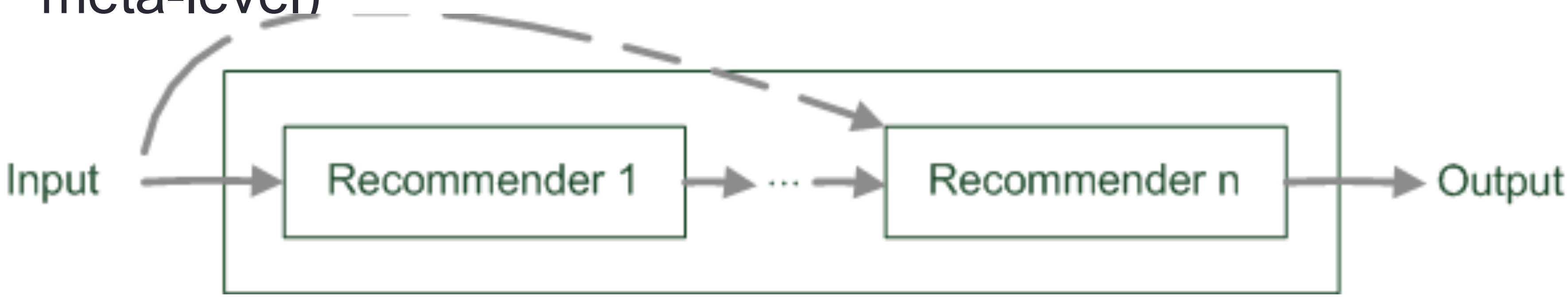
Parallelized hybridization design

- Output of several existing implementations combined
- Least invasive design
- Weighting or voting scheme applied
  - Weights can be learned dynamically



Pipelined hybridization designs

- One recommender system pre-processes some input for the subsequent one
  - Cascade
  - Meta-level
- Refinement of recommendation lists (cascade)
- Learning of model (e.g. collaborative knowledge-based meta-level)



TAGS/PREFS/ EXPLANATIONS

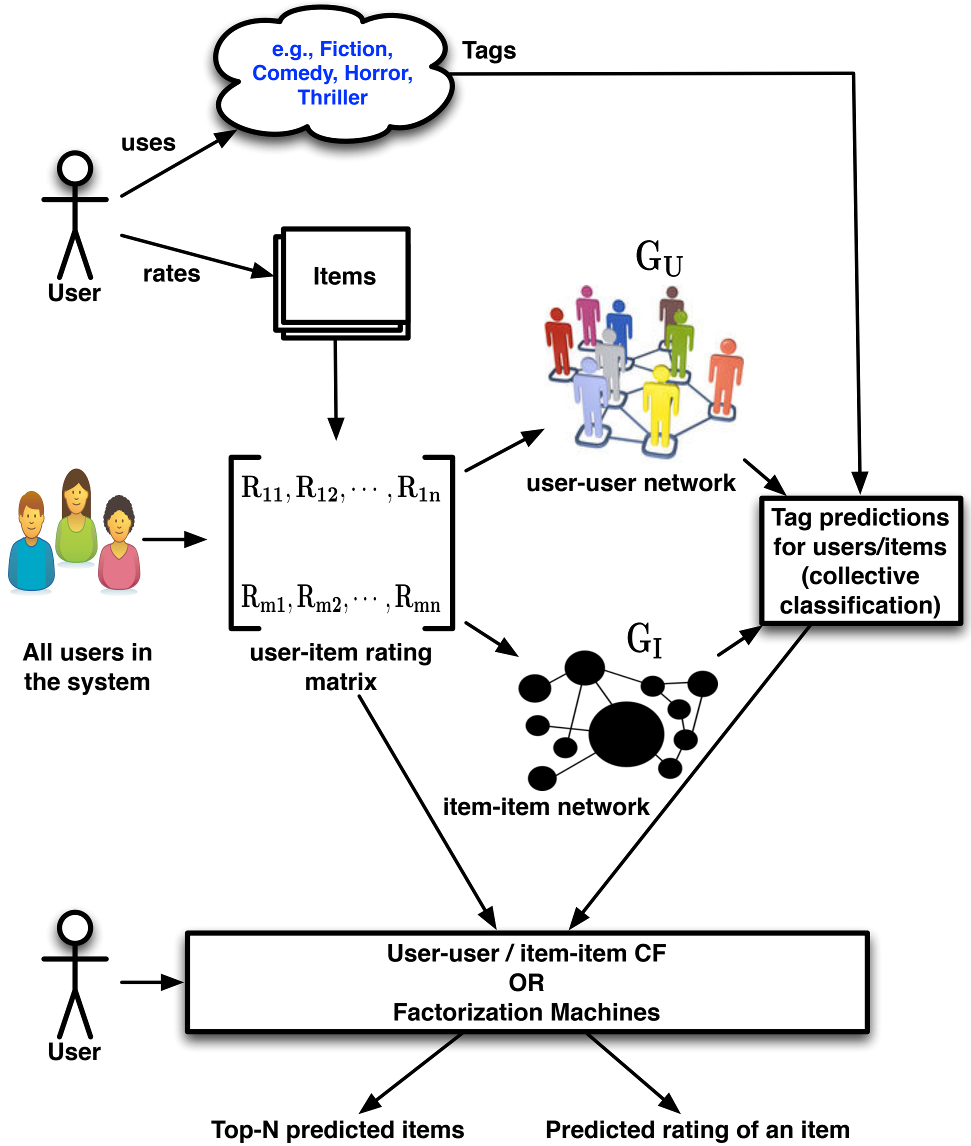
Explanations in recommender systems

Motivation

- “The digital camera *Profishot* is a must-buy for you because . . . .”
- Why should recommender systems deal with explanations at all?
- The answer is related to the two parties providing and receiving recommendations:
  - A selling agent may be interested in promoting particular products
  - A buying agent is concerned about making the right buying decision

Tag-based Recommender Systems

[Saha, Rangwala, Domeniconi. SDM 2015 (submitted)]



Pre-condition:
(i) preference tags
for some users
(ii) item descriptor
tags for some items
are available
as training data

Collaborative
Filtering Recommender
System (CF)

Results

Sample Tags in MovieLens data: rosebud, Johnny Depp, surreal, drugs, space, fantasy

Sample Tags in LibraryThing data: classics, series, short stories, fantasy, humor, non-fiction

Sample Tags in TripAdvisor data: Solo, Friends, Couple, Family

Dataset	FM	IT-FM	IT-FM-G	UIT-FM	UIT-FM-G
TripAdvisor	0.8704 ± 0.0077	0.8480 ± 0.0043	0.8310 ± 0.0047	✓ 0.8429 ± 0.0044	0.8312 ± 0.0063
MovieLens	0.5741 ± 0.0013	✓ 0.5706 ± 0.0016	0.5674 ± 0.0015	0.5751 ± 0.0019	0.5724 ± 0.0011
LibraryThing	0.6128 ± 0.0014	✓ 0.6045 ± 0.0015	0.5959 ± 0.0012	0.6085 ± 0.0019	0.5985 ± 0.0012

MAE for Tag based Methods

Dataset	UIT-FM vs FM	UIT-FM-G vs FM	IT-FM vs FM	IT-FM-G vs FM
TripAdvisor	≈ 0	≈ 0	≈ 0	≈ 0
MovieLens	0.7645	0.4202	0.1283	≈ 0
LibraryThing	0.1059	≈ 0	≈ 0	≈ 0

p-value for Tag based Methods vs Baselines w.r.t. MAE

Tag based matrix factorization methods are better for datasets with meaningful tags

If total number of tags are less, then performance is better

Other Directions

- Temporal ?

- Contextual ?
- Time of the Day, Day of the Week

- ACM RecSYS Competition on Click-Through Prediction.

- Great Fun Projects!