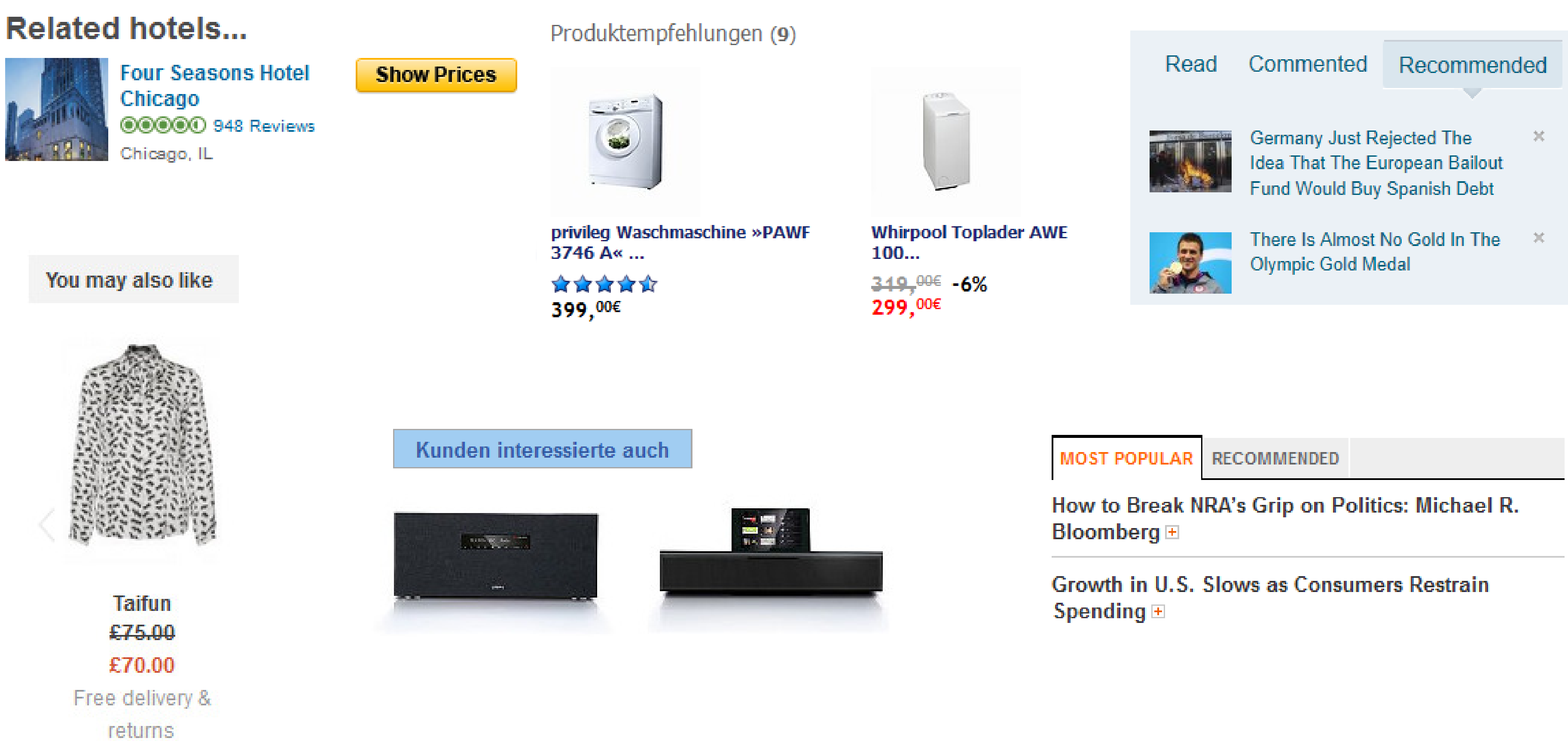
Recommender Systems
An introduction

Dietmar Jannach, TU Dortmund, Germany
Slides presented at PhD School 2014, University Szeged, Hungary

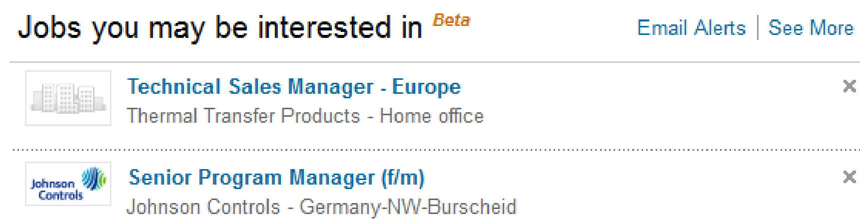
dietmar.jannach@tu-dortmund.de

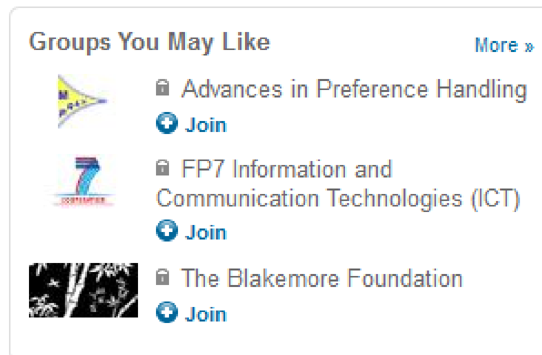
Recommender Systems

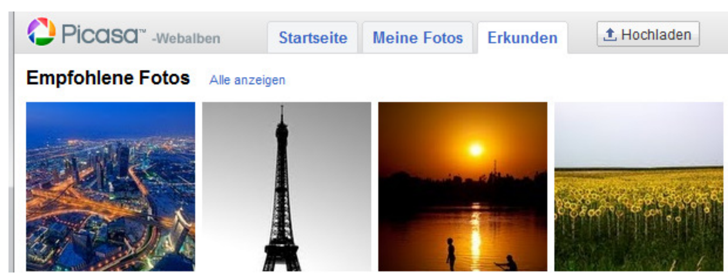
In e-commerce



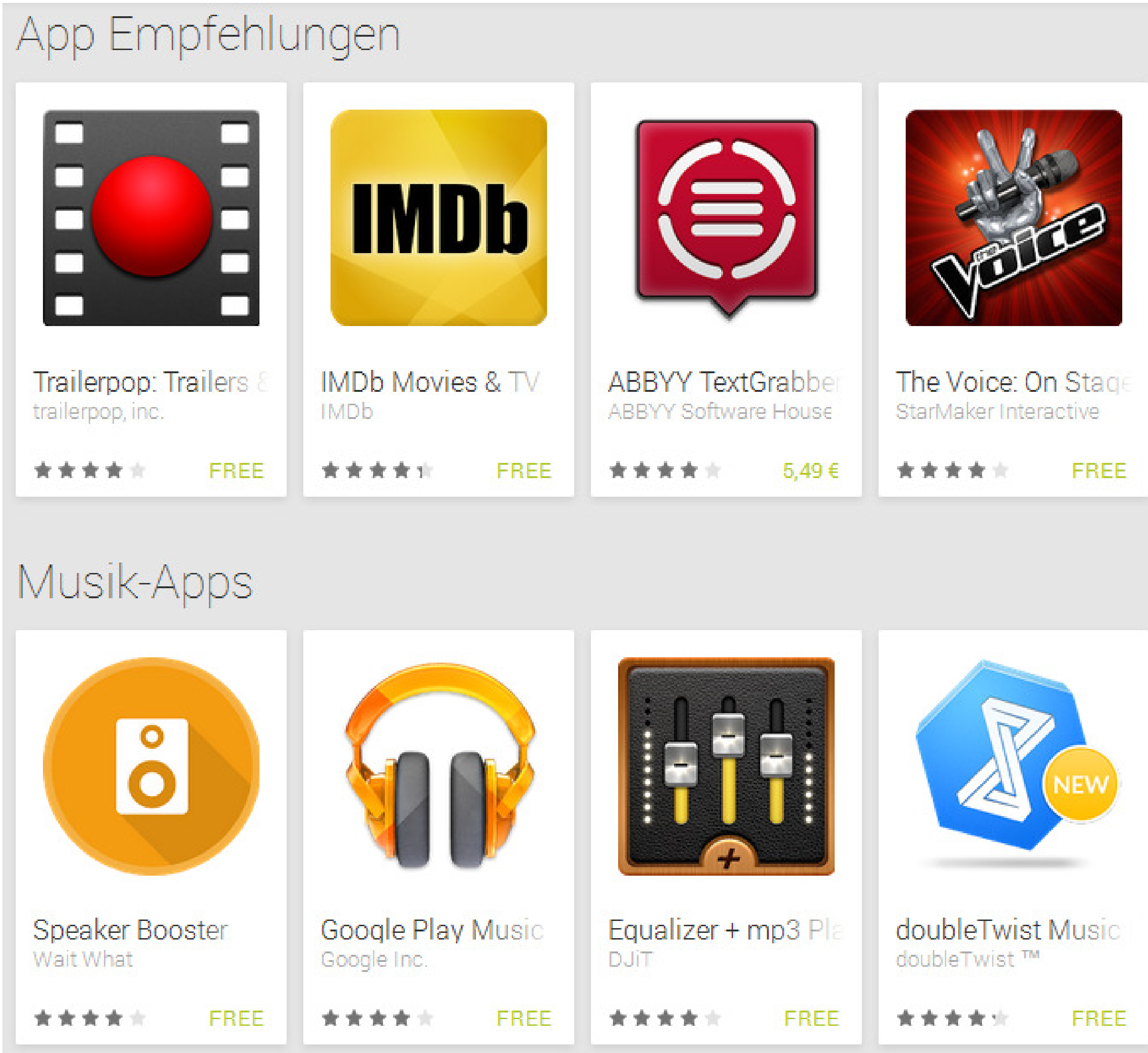
In Social Media





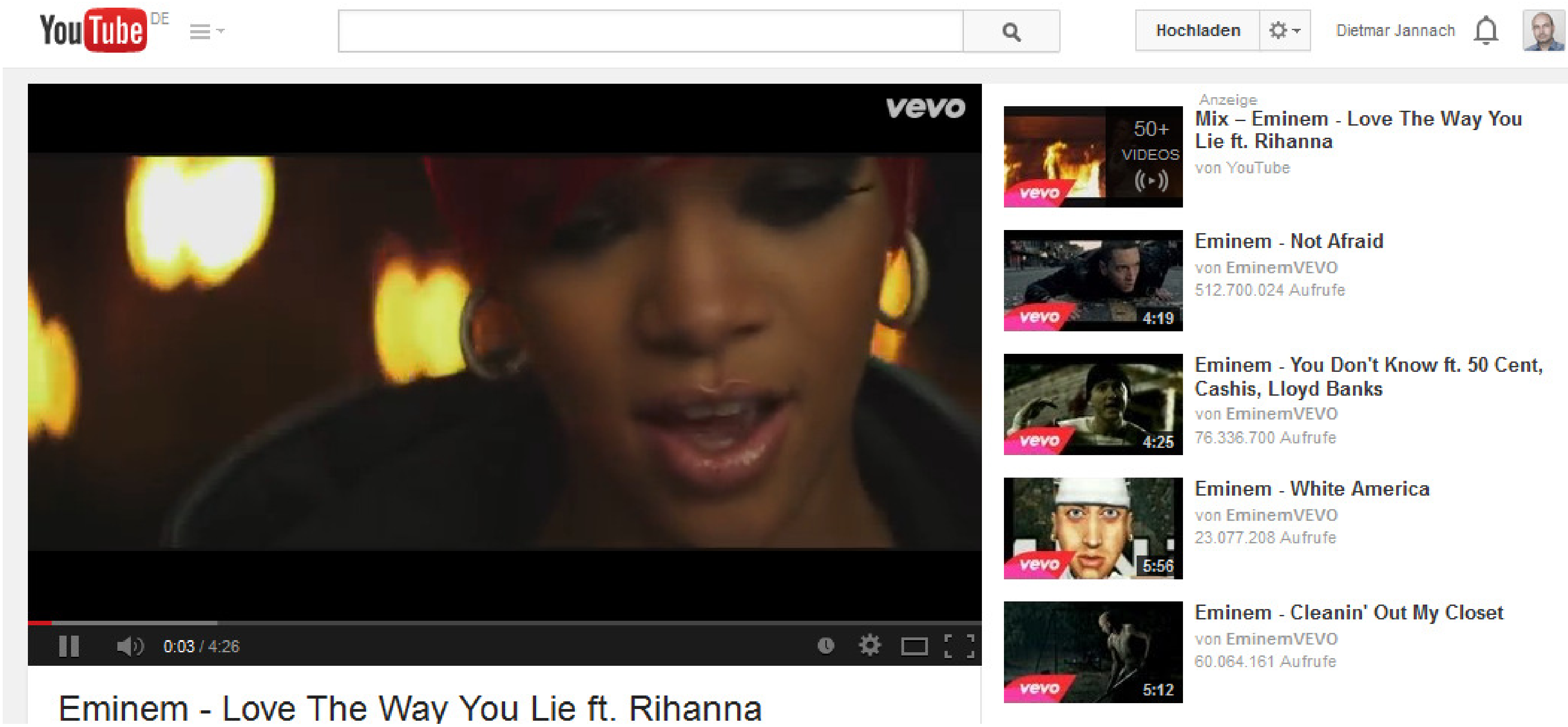


Apps ..



App Empfehlungen
IMDb
Trailerpop: Trailers &
trailerpop, inc.
FREE
IMDb Movies & TV
IMDb
FREE
ABBYY TextGrabber
ABBYY Software House
5,49 €
The Voice: On Stage
StarMaker Interactive
FREE
Musik-Apps
NEW
Speaker Booster
Wait What
FREE
Google Play Music
Google Inc.
FREE
Equalizer + mp3 Pla
DJiT
FREE
doubleTwist Music
doubleTwist ™
FREE

Entertainment



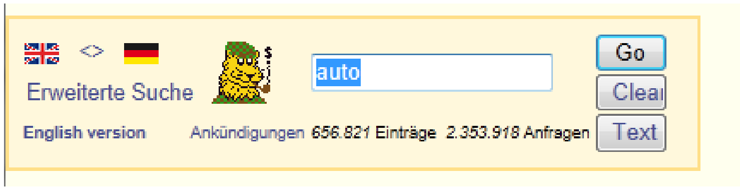
And even more

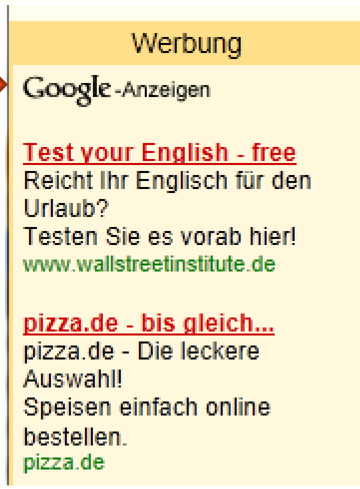
Tags

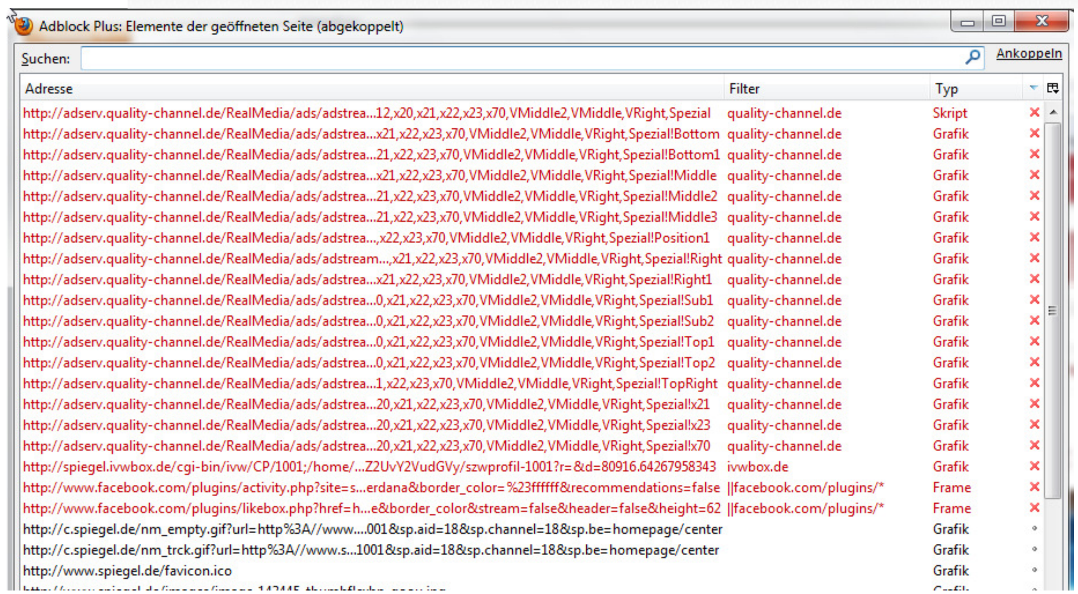
outback • australia • silverton • broken hill • vw • beetle • volkswagen • car • sky • rusty

Limitations

Computational advertising …







About this course

- Will give you an introduction to the field of Recommender Systems
  - How can you compute recommendations?
  - How can we know that the recommendations are good?
  - Current limitations and developments in research
  - Case studies
- Doing a PhD in the field?
  - Emerging topics
  - Publication outlets
- Organization
  - Lectures (morning session), exercises (afternoon session)

About me

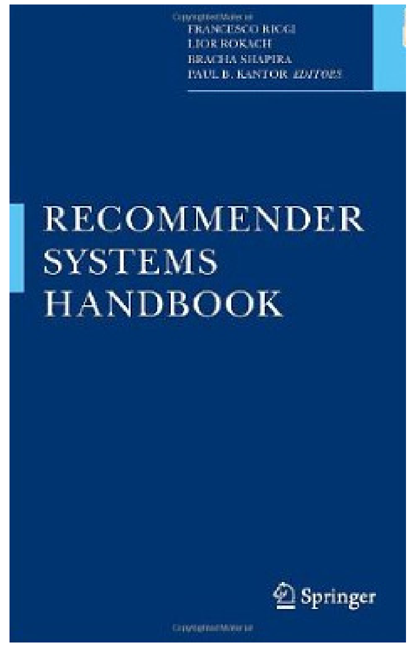
- Professor in Computer Science
  - At TU Dortmund, Germany
- Current research areas
  - Recommender Systems
  - Errors in spreadsheets
- Other topics
  - Artificial Intelligence
  - Web Mining
  - Proucct configuration / Constraints
  - …

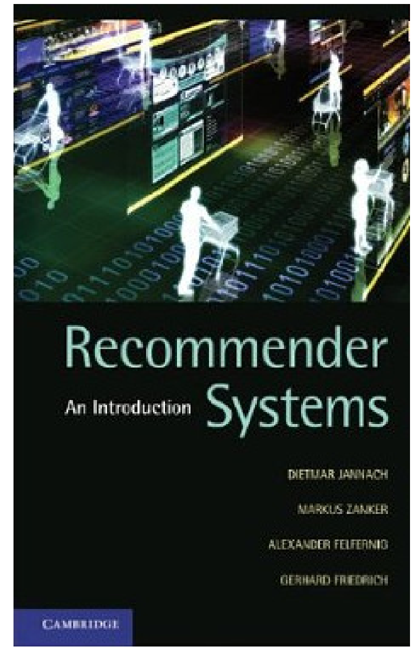
technische universität dortmund

About you

Recommended Reading

- Books
  - Introduction
  - Handbook
- Papers …
  - ACM Conference on Recommender Systems
  - WWW, SIGIR, ICDM, KDD, UMAP, CHI, …
  - Journals on Machine Learning, Data Mining, Information Systems, Data Mining, User Modeling, Human Computer Interaction, …
- Special issues on different topics published





Why using Recommender Systems?

- Value for the customer
  - Find things that are interesting
  - Narrow down the set of choices
  - Help me explore the space of options
  - Discover new things
  - Entertainment
  - …

- Value for the provider
  - Additional and probably unique personalized service for the customer
  - Increase trust and customer loyalty
  - Increase sales, click trough rates, conversion etc.
  - Opportunities for promotion, persuasion
  - Obtain more knowledge about customers
  - …

Real-world check

- Myths from industry
  - Amazon.com generates X percent of their sales through the recommendation lists (30 < X < 70)
  - Netflix (DVD rental and movie streaming) generates X percent of their sales through the recommendation lists (30 < X < 70)
- There must be some value in it
  - See recommendation of groups, jobs or people on LinkedIn
  - Friend recommendation and ad personalization on Facebook
  - Song recommendation at last.fm
  - News recommendation at Forbes.com (plus 37% CTR)
- In academia
  - A few studies exist that show the effect
    - increased sales, changes in sales behavior

Outline of the lecture

- Introduction
- How do recommender systems (RS) work ?
  - Collaborative filtering
  - Content-based filtering
  - Knowledge-based recommenders
  - Hybrid Systems
- How do they influence users and how do we measure their success?
  - Different tvaluation designs
  - Case study
- Selected topics in recommender systems
  - Explanations, Trust, Robustness, Multi-criteria ratings, Context-aware recommender systems

Definition – Problem domain

- Recommendation systems (RS) help to match users with items
  - Ease information overload
    - How many books on Amazon?
    - How many tracks on iTunes?
  - 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]

An often-cited problem characterization

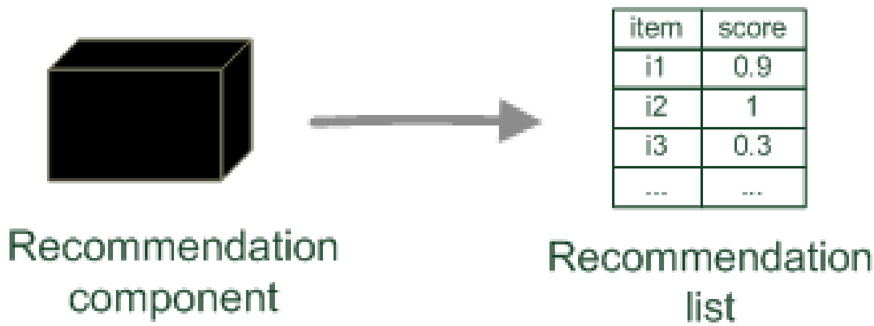
(Adomavicius & Tuzhilin, TKDE, 2005)

- Given
  - The profile of the "active" user and possibly some situational context
- Compute
  - A relevance (ranking) score for each recommendable item

- The profile …
  - … can include past user ratings (explicit or implicit), demographics and interest scores for item features
- The problem …
  - … is to learn a function that predicts the relevance score for a given (typically unseen) item

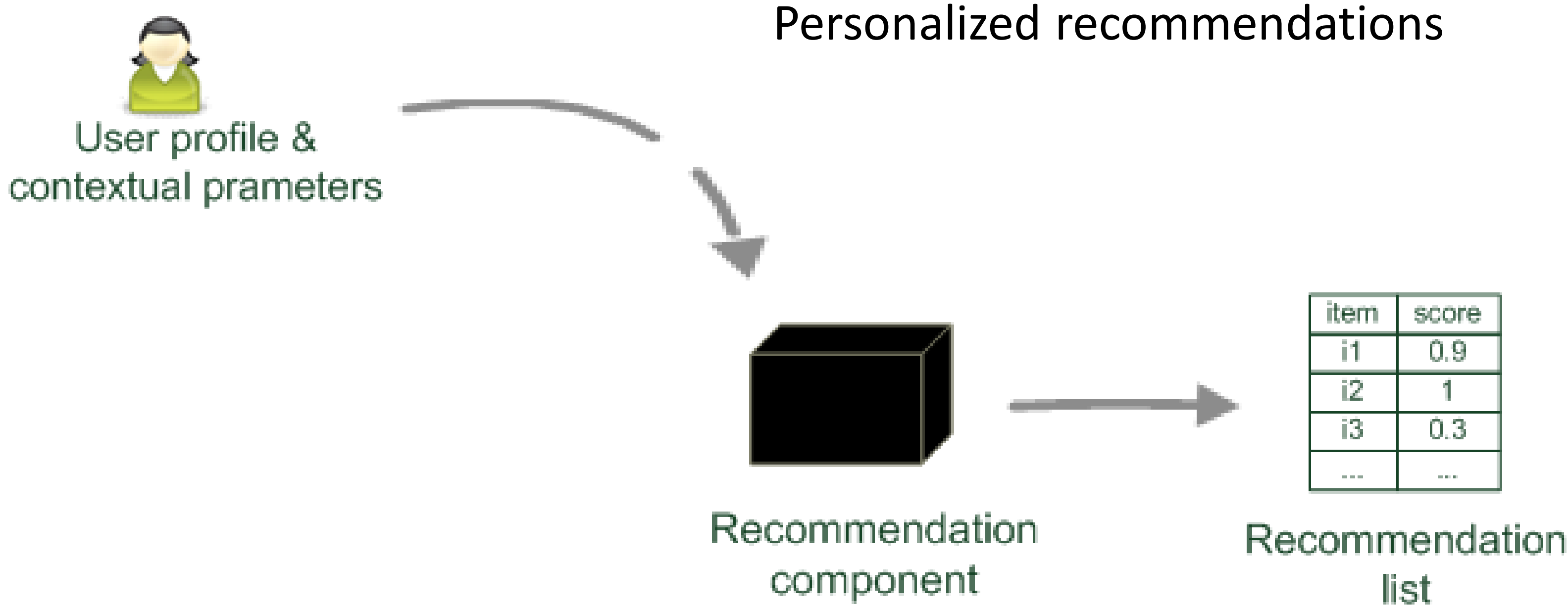
Paradigms of recommender systems

Recommender systems reduce information overload by estimating relevance

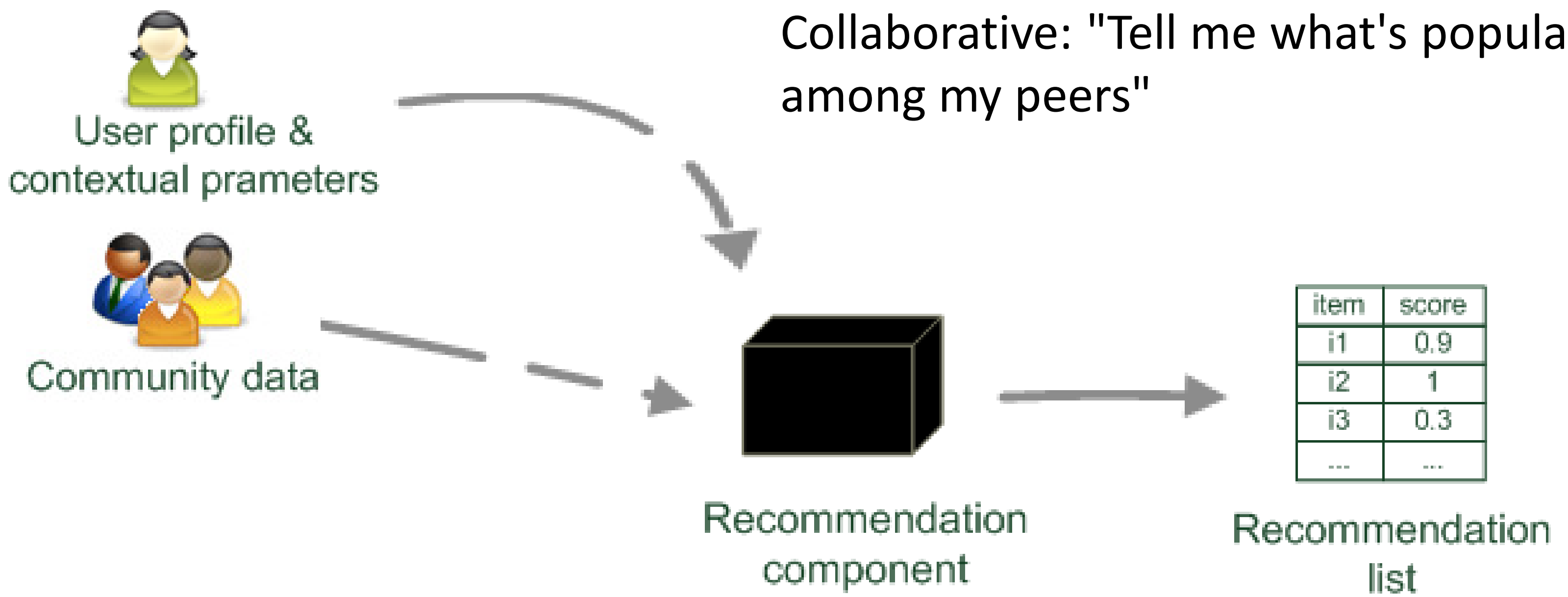


Paradigms of recommender systems

Personalized recommendations



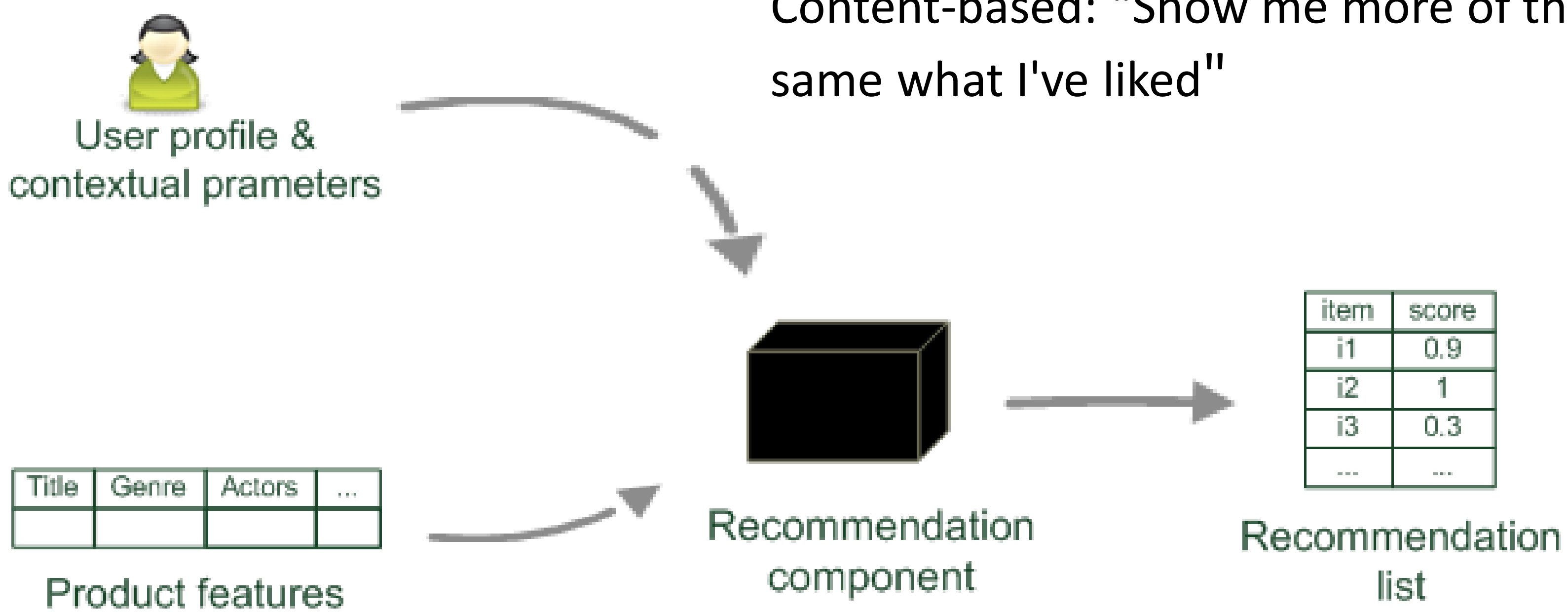
Paradigms of recommender systems



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

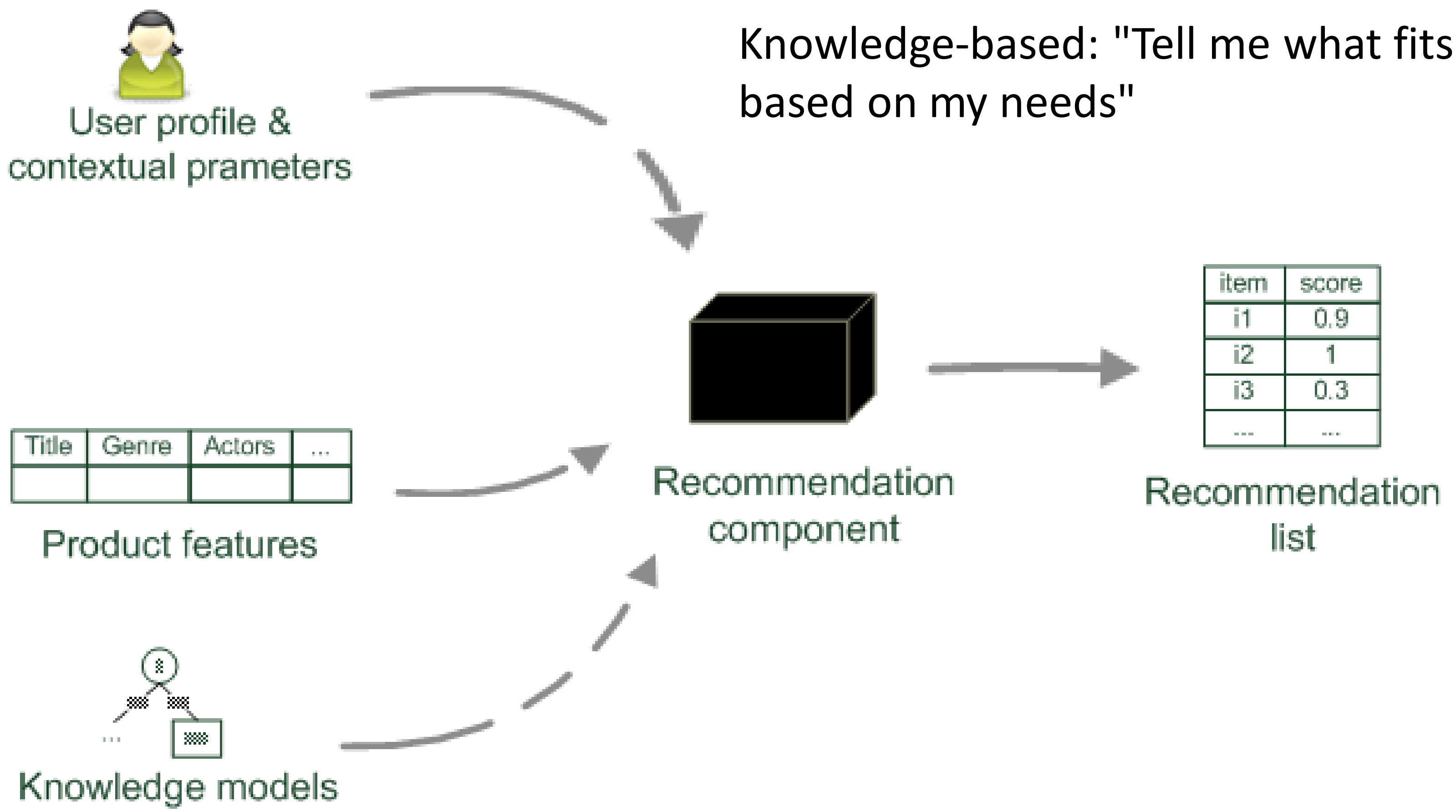
Paradigms of recommender systems

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



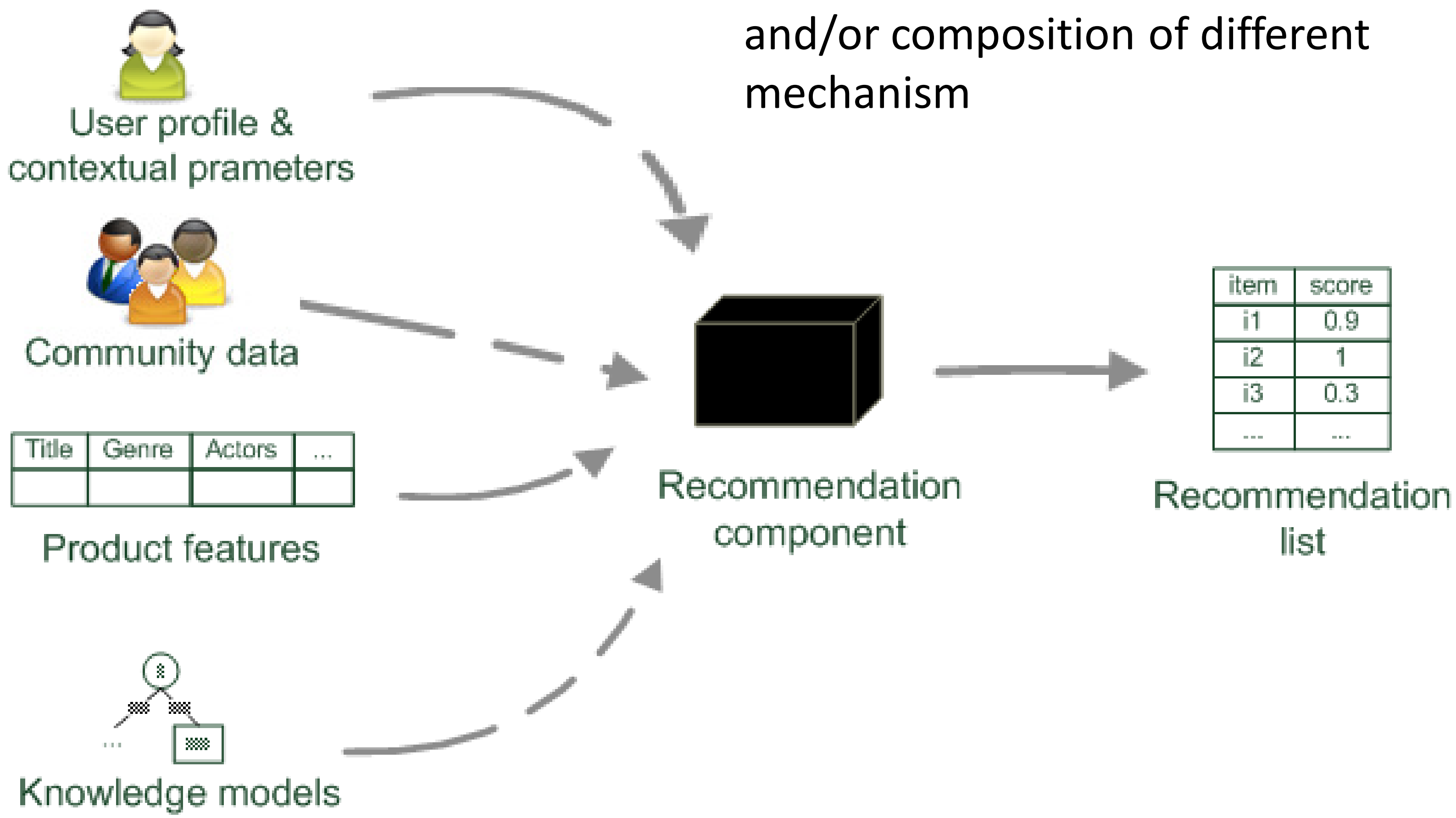
Paradigms of recommender systems

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

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 preferences of a community to recommend items
- Basic assumption and idea
  - Users give ratings to catalog items (implicitly or explicitly)
  - Patterns in the data help me predict the ratings of individuals, i.e., fill the missing entries in the rating matrix, e.g.,
    - there are customers with similar preference structures,
    - there are latent characteristics of items that influence the ratings by users
    - …

1992: *Using collaborative filtering to weave an information tapestry*

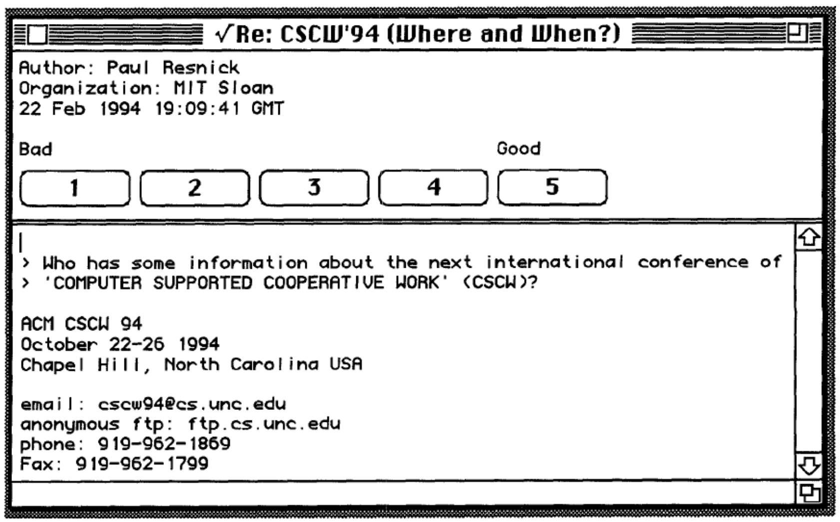
(D. Goldberg et al., Comm. of the ACM)

- Basic idea:
  - Eager readers read all docs immediately, casual readers wait for the eager readers to annotate
- Experimental mail system at Xerox Parc
  - Records reactions of users when reading a mail
- Users are provided with personalized mailing list filters instead of being forced to subscribe
  - Content-based filters (topics, from/to/subject…)
  - Collaborative filters
    - "Mails to [all] which were replied by [John Doe] and which received positive ratings from [X] and [Y]."

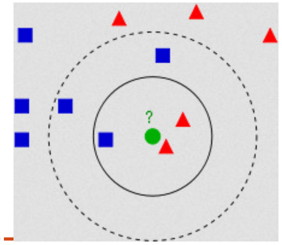
1994: *GroupLens: an open architecture for collaborative filtering of netnews*

(P. Resnick et al., ACM CSCW)

- Tapestry system does not aggregate ratings and requires knowing each other
- Basic idea of GroupLens:
  - People who agreed in their subjective evaluations in the past are likely to agree again in the future
- Builds on newsgroup browsers with rating functionality



Nearest-neighbors (kNN)



- A "pure" CF approach and traditional baseline
  - Uses a matrix of (explicit) ratings provided by the community as inputs
  - Returns a ranked list of items based on rating predictions
- Solution approach
  - Given an "active user" (Alice) and an item I not yet seen by Alice
  - Estimate Alice's rating for this item based on like-minded users (peers)
- Assumptions
  - If users had similar tastes in the past they will have similar tastes in the future
  - User preferences remain stable and consistent over time

Questions to answer...

1) How to determine the similarity of two users?
2) How do we combine the ratings of the neighbors to predict Alice's rating?
3) Which/how many neighbors' opinions to consider?

	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

1 Determining similarity

- A popular measure: Pearson's correlation coefficient

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

a, b : users
$r_{a,p}$: rating of user a for item p
P : set of items, rated both by a and b
$\overline{r_a}, \overline{r_b}$: user's average ratings

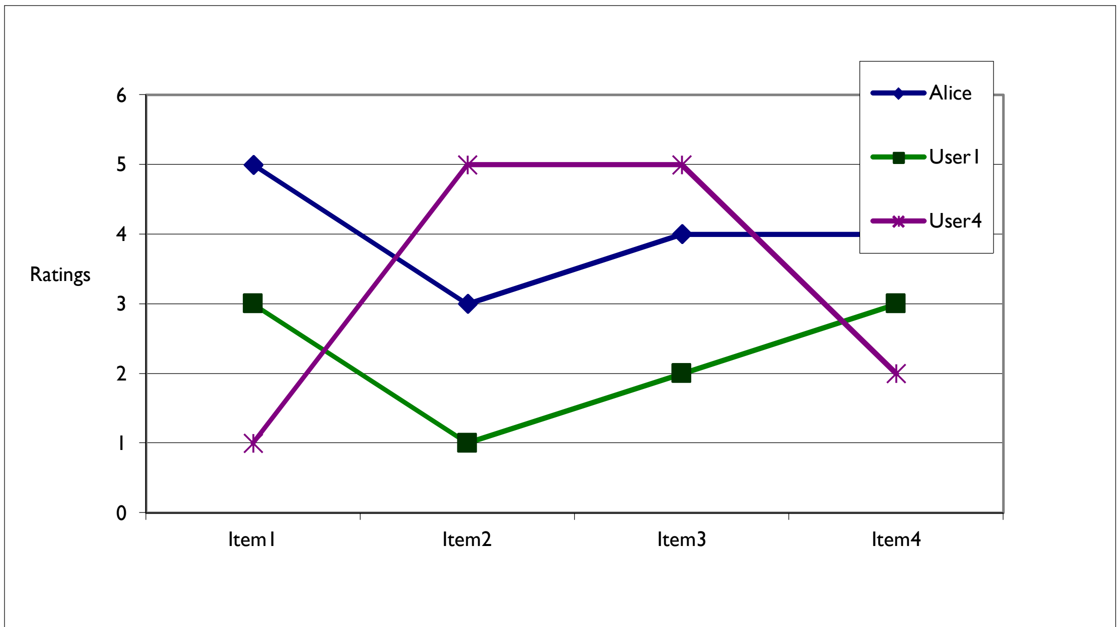
Possible similarity values between -1 and 1;

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

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

Pearson correlation

- Takes differences in rating behavior into account



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

2 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 with as a weight
  - Add/subtract the neighbors' bias from the active user's average and use this as a prediction
- How many neighbors?
  - Only consider positively correlated neighbors (or higher threshold)
  - Can be optimized based on data set
  - Often, between 50 and 200

Improved kNN recommendations
(Breese et al., UAI, 1998)

- 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

kNN considerations

- Very simple scheme leading to quite accurate recommendations
  - Still today often used as a baseline scheme
- Possible issues
  - Scalability
    - Thinking of millions of users and thousands of items
    - Pre-computation of similarities possible but potentially unstable
    - Clustering techniques are often less accurate
  - Coverage
    - Problem of finding enough neighbors
    - Users with preferences for niche products

2001: *Item-based collaborative filtering recommendation algorithms*

B. Sarwar et al., WWW 2001

- 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

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)

Using (adjusted) cosine similarity

- 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

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

  - 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

Slope One predictors

(Lemire and Maclachlan, 2005)

- Idea of Slope One predictors:
  - Based on a popularity differential between items for users
- Example:
  - p(Alice, Item5) = 2 + (2-1) = 3

	Item1	Item5
Alice	2	?
User1	1	2

- Basic scheme:
  - Take the average of these differences of the co-ratings to make the prediction
  - Different variants proposed
- In general: find a function of the form f(x) = x + b
  - That is why the name is "Slope One"
- Can be computationally complex

RF-Rec predictors (Gedikli et al. 2011)

- Idea: Take rating frequencies into account for computing a prediction
- Basic scheme: $\hat{r}_{u,i} = \arg\max_{v \in R} f_{user}(u, v) * f_{item}(i, v)$
  - R: Set of all rating values, e.g., $R = \{1,2,3,4,5\}$ on a 5-point rating scale
  - $f_{user}(u, v)$ and $f_{item}(i, v)$ basically describe *how often* a rating v was assigned by user u and to item i resp.
- Example:

	Item1	Item2	Item3	Item4	Item5
Alice	1	1	?	5	4
User1	2		5	5	5
User2			1	1	
User3		5	2		2
User4	3		1	1	
User5	1	2	2		4

- p(Alice, Item3) = 1
- Extended with optimized weighting scheme

Memory- and model-based approaches

- kNN methods are often 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

Model-based approaches

- Variety of techniques proposed in recent years, e.g.,
  - Matrix factorization techniques
    - 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
    - Regression-based techniques, deep neural networks, …
- Costs of pre-processing
    - Usually not discussed
    - Incremental updates possible – algorithms exist

A data mining approach: 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"
- Simple co-occurrences (conditional probabilities)
  - "Customers who bought/views, also bought .."

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

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 ..
- Can be used for binary/unary ratings and implicit feedback
- Different (distributed) algorithms available
  - FP-Growth, CFP-Growth, PFP-Growth

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(Item5=1 | X), where
    - X = Alice's previous ratings = (Item1 =1, Item2=3, Item3= …)
  - Can be estimated based on Bayes' Theorem

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

  - Assumption: Ratings are independent (?)

Calculation of probabilities (simplistic)

	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= …)

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

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

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

Practical probabilistic approaches

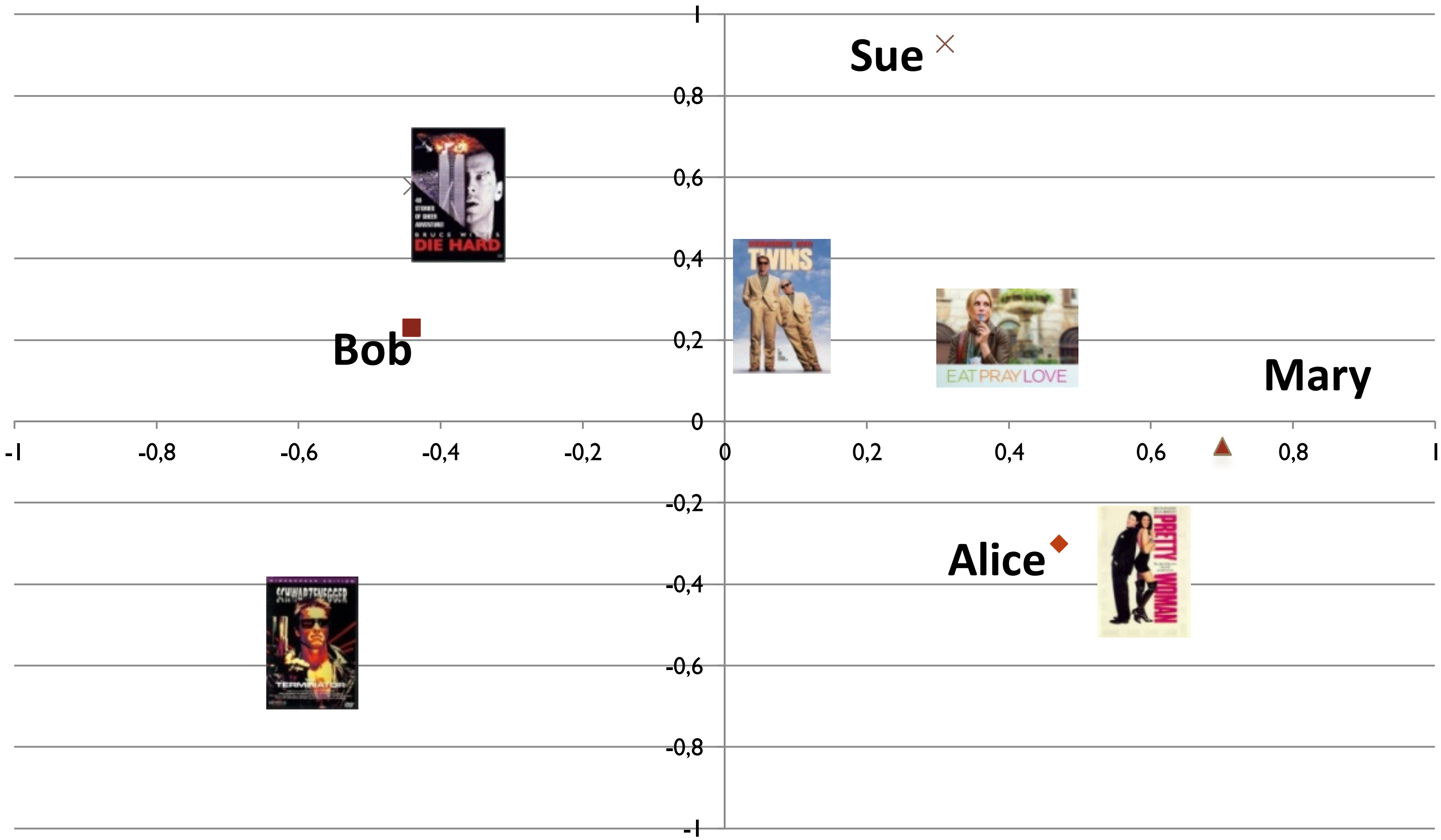
- Use a cluster-based approach (Breese et al. 1998)
  - assume users fall into a small number of subgroups (clusters)
  - Make predictions based on estimates
    - probability of Alice falling into cluster c
    - probability of Alice liking item i given a certain cluster and her previous ratings
    - $P(C = c, v_1, \ldots, v_n) = P(C = c) \prod_{i=1}^{n} P(v_i | C = c)$
  - 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

2000: *Application of Dimensionality Reduction in Recommender Systems*

(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,…

The "latent factor space"



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 Σ 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 Σ (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

- U and V correspond to the latent user and item factors

- 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

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

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

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

- Matrix factorization
  - 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

2006 "Funk-SVD" and the Netflix prize

(S. Funk, Try this at home)

- Netflix announced a million dollar prize
  - Goal:
    - Beat their own "Cinematch" system by 10 percent
    - Measured in terms of the Root Mean Squared Error
      - (evaluation aspects will discussed later on)
  - Effect:
    - Stimulated lots of research
- Idea of SVD and matrix factorization picked up again
  - S. Funk (pen name)
    - Use fast gradient descent optimization procedure
    - http://sifter.org/~simon/journal/20061211.html

Learn the weights in iterative approach

- Start with small initial weights
- Repeat
  - Make prediction with current model
  - Adapt the weights incrementally
    - learning rate as a hyperparameter
  - Stop after n iterations

Q

	Factor 1	Factor 2	Factor 3	...	Factor n
User1	0,1	0,3	0,001		0,2
User2	0,1	0,23	0,3		0,1
User3	0,5	0,4	0,4		0,1
...					
User n	0,1	0,2	0,7		0,2

P

Item 1	0,1	0,33	0,2		0,1
Item 2	0,5	0,23	0,01		0,8
..					
Item n	0,1	0,23	0,3		0,3

2008: *Factorization meets the neighborhood: a multifaceted collaborative filtering model*

(Y. Koren, ACM SIGKDD)

- Combines 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 similar tems
  - Combination in one prediction single function
    - Includes user- and item bias, considers who rated what
    - Add penalty (regularization) for high values to avoid over-fitting

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

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

Generalization: An optimization problem

- 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 and Factorization Machines, Deep Neural Networks, …
  - Netflix Prize winner:
    - Combine a large number of predictors in ensemble method

A prediction with 0% error!

- Past profile
  - You liked Star Wars and
  - you gave five stars to Star Wars I to Star Wars III
- My prediction is that you
  - will give five stars to Star Wars III to Star Wars Infinity
  - I recommend more Star Wars movies

- Exact rating predictions might not enough
  - No surprise
    - no extra sales and limited value
  - No variety in recommendations …

Rating prediction & Item recommendation

- 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
- Ranking approaches
  - "Learning to rank"
    - Recent interest in ranking techniques
    - Optimize according to a (proxy of a) given rank evaluation metric

Explicit and implicit ratings

- Explicit ratings
  - Most commonly used (1 to 5, 1 to 7 Likert response scales)
  - Typically only one rating per user and item, including time-stamp
- Some research topics
  - Data sparsity
    - Users not always willing to rate many items
    - How to stimulate users to rate more items?
  - Which items have (not) been rated?
    - Ratings not missing at random
  - Optimal granularity of scale
    - Indication that 10-point scale is better accepted in movie domain
    - An even more fine-grained scale was chosen in the Jester joke recommender
  - Multidimensional ratings
    - multiple ratings per movie (acting, directing, …)

Explicit and implicit ratings

- Implicit ratings (feedback)
  - Typically collected by the web shop or application in which the recommender system is embedded
    - Clicks, page views, time spent on some page, demo downloads …
    - Multiple events over time
  - Can be collected constantly and do not require additional efforts from the side of the user
- Research topics
  - Correct interpretation of the (strength of the) action
    - Buy something for a friend, accidental clicks
    - How to interpret shopping cart actions (recommend or not?)
  - Huge amounts of data to be processed
  - Algorithmic questions
    - Combination with explicit ratings
      - e.g., Koren's SVD++ method
    - Specific algorithms (e.g., Bayesian Personalized Ranking)

Data sparsity – cold start situations

- How to recommend new items? What to recommend to new users?
- A problem even on large platforms
  - e.g., hotel review platforms – domain specific issues
- 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
- Alternatives
  - Use better algorithms (beyond nearest-neighbor approaches)
  - Exploit additional information sources, e.g., Social Web data
- Example:
  - In nearest-neighbor approaches, the set of sufficiently similar neighbors might be too small to make good predictions
  - Assume "transitivity" of neighborhoods

Example algorithm for sparse datasets

- Recursive CF (Zhang and Pu 2007)
  - Assume there is a very close neighbor n of u who however has not rated the target item i yet.
  - Idea:
    - Apply CF-method recursively and predict a rating for item i for the neighbor
    - Use this predicted rating instead of the rating of a more distant direct neighbor

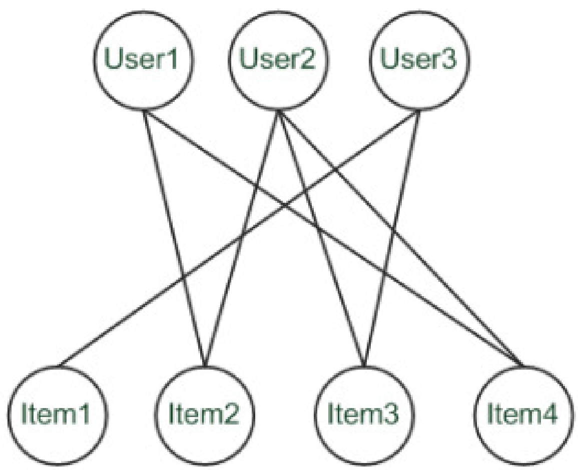
	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	?
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = 0.85

Predict rating for User1

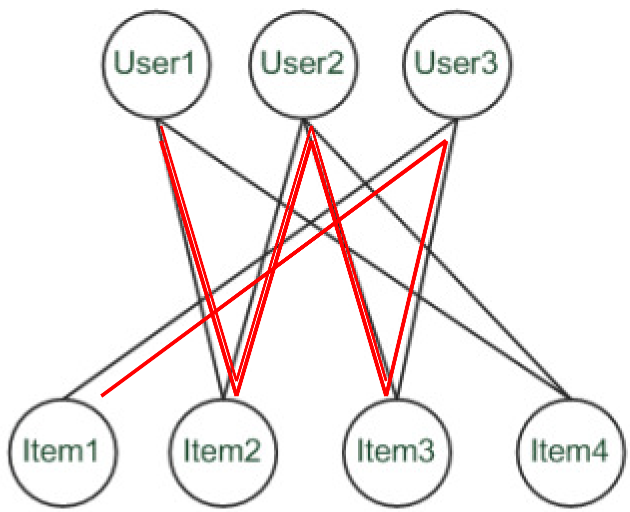
A graph-based approach

- Spreading activation (Huang et al. 2004)
  - Exploit the supposed "transitivity" of customer tastes and thereby augment the matrix with additional information
  - Assume that we are looking for a recommendation for *User1*
  - Standard CF approach:
    - *User2* will be considered a peer for *User1* because they both bought *Item2* and *Item4*
    - *Item3* will be recommended to *User1* because the nearest neighbor, *User2*, also bought or liked it



A graph-based approach

- Spreading activation (Huang et al. 2004)
  - Standard CF approaches:
    - paths of length 3 will be considered
    - *Item3* is relevant for *User1* because there exists a three-step path (*User1–Item2–User2–Item3*) between them
  - Here:
    - The idea is to also consider longer paths (indirect associations) to compute recommendations
    - Using path length 5, for instance



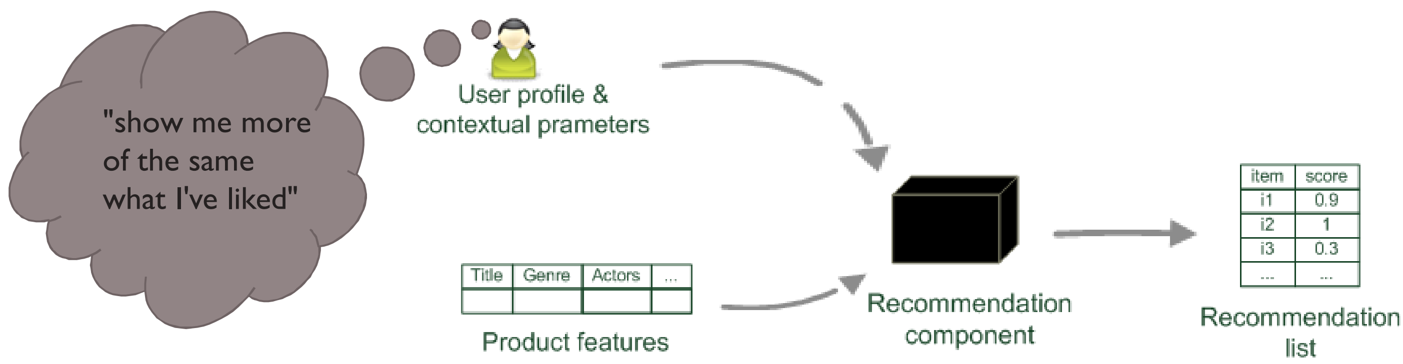
Summary CF approaches

- Operate on the basis of explicit or implicit feedback of a a user community
  - Well-understood, lots of algorithms
  - Works in practice
    - in particular for quality-and-taste domains
  - No information about the items required
- Challenges
  - Cold start and data sparsity issues
  - Scalability can be an issue
  - Often no explanations possible
  - Not applicable in every domain
    - e.g., when specific, short-term user preferences have to be respected or there are complex products (cameras, cars, …)

CF tools and libraries

- Some open source solutions exist
  - MyMediaLite
    - Implements wide range of modern algorithms
    - Implemented in C#
  - LensKit
    - Modular framework built in Java
    - Provided by the GroupLens research group
  - PREA
    - Java-based library of recent CF algorithms
  - Apache Mahout, RapidMiner, Apache Spark + MLib
    - Implement learning algorithms usable for recommenders
    - Mahout: distributed algorithms on Hadoop
  - Recommender101
    - Java-based framework, several algorithms and metrics

Content-based filtering



Content-based Filtering

- Again:
  - Determine preferences of user based on past behavior
- This time, however:
  - Look at what the current user liked (purchased, viewed, …)
  - Estimate the user's preference for certain item features
    - e.g., genre, authors, release date, keywords in the text
  - Alternative preference acquisition
    - ask the user, look at recently viewed items

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
  - Unstructured: free-text description.

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

Content representation and item similarities

- Represent items and users in the same way

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	Brunonia, Barry, Ken Follett	Paperback	25.65	Detective, murder, New York

- A simple method
  - Compute the similarity of an unseen item with the user profile based on the keyword overlap (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:
    - 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 calculation

- Given a keyword i and a document j
- $TF(i,j)$
  - term frequency of keyword i in document j
  - Term frequency is relative to most frequent term z in document j

$$\boldsymbol{TF(i,j) = \frac{Frequency(i,j)}{max_z\, Frequency(z,j)}}$$

- $IDF(i)$
  - inverse document frequency calculated as $\boldsymbol{IDF(i) = log\frac{N}{n(i)}}$
    - N : number of all recommendable documents
    - $n(i)$: number of documents from N in which keyword i appears
- $TF - IDF$
  - is calculated as: $\boldsymbol{TF\text{-}IDF(i,j) = TF(i,j) * IDF(i)}$
- Normalization
  - Vector of length 1

$$\boldsymbol{TF - IDF(i,j) = \frac{TF - IDF(i,j)}{\sqrt{\sum_s TF - IDF(s,j)^2}}}$$

Example TF-IDF representation

- Absolute term frequency:
  - Each document is a $\boxed{\text{count vector}}$ in $\mathbb{N}^{|v|}$

	Poem A	Poem B	Poem C
Caesar	232	0	2
Calpurnia	0	10	0
Cleopatra	57	0	0

Vector v with dimension $|v| = 3$

$$TF(i,j) = \frac{Frequency(i,j)}{max_z\, Frequency(z,j)}$$

TF	Poem A	Poem B	Poem C
Caesar	1	0	1
Calpurnia	0	1	0
Cleopatra	0.24	0	0

Example TF-IDF representation

	Poem A	Poem B	Poem C
Caesar	232	0	2
Calpurnia	0	10	0
Cleopatra	57	0	0

$$IDF(i) = log\frac{N}{n(i)}$$

IDF	Poem A	Poem B	Poem C
Caesar	0.58	0	0.58
Calpurnia	0	1.58	0
Cleopatra	1.58	0	0

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

TF-IDF	Poem A	Poem B	Poem C
Caesar	0.58	0	1
Calpurnia	0	1.58	0
Cleopatra	0.39	0	0

$$TF - IDF(i,j) = \frac{TF - IDF(i,j)}{\sqrt{\sum_s TF - IDF(s,j)^2}}$$

Norm. TF-IDF	Poem A	Poem B	Poem C
Caesar	0.83	0	1
Calpurnia	0	1	0
Cleopatra	0.55	0	0

Given numbers are not correct here…

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
- Tuning of representation
  - Logarithmic instead of linear TF count

Improving the vector space model

- Use lexical knowledge, use more elaborate methods for feature selection
  - Remove words that are not relevant in the domain
- Detection of phrases/n-grams
  - 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

Comparing the vectors (users/items)

- Usual similarity metric to compare vectors: Cosine similarity (angle)
  - Cosine similarity is calculated based on the angle between the vectors
  - Compensates for the effect of different document lengths

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

- Query "Caesar Calpurnia"
  - Similarity between query and documents

Norm. TF-IDF	Antony and Cleopatra	Julius Caesar	Hamlet	Query
Caesar	0.83	0	1	0.35
Calpurnia	0	1	0	0.94
Cleopatra	0.55	0	0	0
Similarity to query	0.29	**0.94**	0.35	1

Recommending items

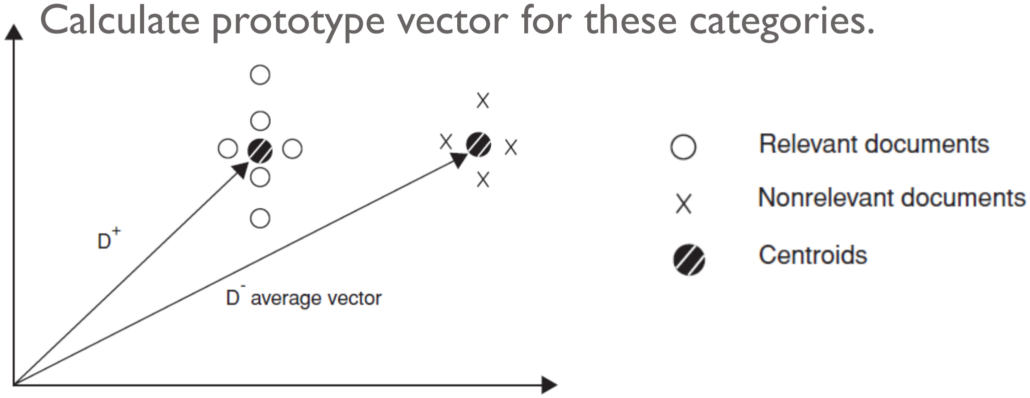
- Item recommendation: nearest neighbors
  - Given a set of documents D already rated by the user (like/dislike)
    - Either explicitly via user interface
    - Or implicitly by monitoring user's behavior
  - Find the n nearest neighbors of a not-yet-seen item i in D
    - Use similarity measures (like cosine similarity) to capture similarity between two documents
- Rating predictions
  - 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 $\Rightarrow$ 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

Rocchio's method

- Retrieval quality depends on individual capability to formulate queries with suitable keywords
- 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

Rocchio details

- Document collections D^+ (liked) and D^- (disliked)
  - Calculate prototype vector for these categories.



  - Computing modified query Q_{i+1} from current query Q_i with:

$$Q_{i+1} = \alpha * Q_i + \beta\left(\frac{1}{|D^+|}\sum_{d^+\in D^+} d^+\right) - \gamma\left(\frac{1}{|D^-|}\sum_{d^-\in D^-} d^-\right)$$

α, β, γ used to fine-tune the feedback

α weight for original query

β weight for positive feedback

γ weight for negative feedback

  - Often only positive feedback is used
  - More valuable than negative feedback

Probabilistic methods

- Recommendation as classical text classification problem
  - long history of using probabilistic methods
- Simple approach:
  - 2 classes: hot/cold
  - Simple Boolean document representation
  - Calculate probability that document is hot/cold based on Bayes theorem

Doc-ID	recommender	intelligent	learning	school	Label
1	1	1	1	0	1
2	0	0	1	1	0
3	1	1	0	0	1
4	1	0	1	1	1
5	0	0	0	1	0
6	1	1	0	0	?

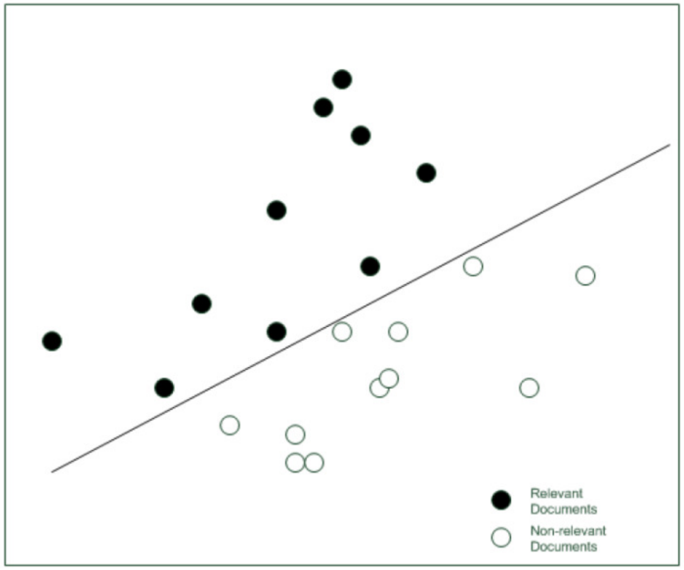
$$
\begin{aligned}
& P(X|Label = 1) \\
&= P(recommender = 1|Label = 1) \\
&\times P(intelligent = 1|Label = 1) \\
&\times P(learning = 0|Label = 1) \\
&\times P(school = 0|Label = 1) \\
&= {}^{3}/_{3} \times {}^{2}/_{3} \times {}^{1}/_{3} \times {}^{2}/_{3} \approx 0.149
\end{aligned}
$$

Improvements

- Side note: Conditional independence of events does in fact not hold
  - "New York", "Hong Kong"
  - Still, good accuracy can be achieved
- Boolean representation simplistic
  - positional independence assumed
  - keyword counts lost
- More elaborate probabilistic methods
  - e.g., estimate probability of term v occurring in a document of class C by relative frequency of v in all documents of the class
- Probabilistic Latent Semantic Analysis
  - Find latent topics within documents (compare Matrix Factorization and SVD methods)
- Other linear classification algorithms (machine learning) can be used
  - Support Vector Machines, ..

Linear classifiers

- Most learning methods aim to find coefficients of a linear model
  - A simplified classifier with only two dimensions can be represented by a line
  - The line has the form $\boldsymbol{w_1 x_1 + w_2 x_2 = b}$
  - x_1 and x_2 correspond to the vector representation of a document (using e.g. TF-IDF weights)
  - w_1, w_2 and b are parameters to be learned
  - Classification of a document based on checking
    $$w_1 x_1 + w_2 x_2 > b$$
  - In n-dimensional space the classification function is $\boldsymbol{\vec{w}^T \vec{x} = b}$



On feature selection

- Process of choosing a subset of available terms
- Different strategies exist for deciding which features to use
  - Feature selection based on domain knowledge and lexical information from WordNet
  - Frequency-based feature selection to remove words appearing "too rare" or "too often"
- Not appropriate for larger text corpora
  - Better to
    - evaluate value of individual features (keywords) independently and
    - construct a ranked list of "good" keywords.
  - Typical measure for determining utility of keywords:

    e.g. X^2, mutual information measure or Fisher's discrimination index

Limitations of content-based 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

Discussion & summary

- Content-based techniques do not require a user community
  - They however require content information
  - Recent new types of "content" information
    - Wikipedia, Linked Data, Social Tags, Social Media posts…
- The presented approaches learn a model of the user's interest preferences based on explicit or implicit feedback
  - Deriving implicit feedback from user behavior can be problematic
- Danger exists that recommendation lists contain too many similar items
  - All learning techniques require a certain amount of training data
  - Some learning methods tend to overfit the training data
- Research focuses on CF methods, in practice, however
  - Content-based methods work well in some domains

A case study – mobile games

- Typical in research
  - Offline evaluation (historical datasets)
  - Optimize accuracy or rank metric
- What about the business value?
  - Nearly no real-world studies
  - Exceptions, e.g., Dias et al., 2008.
    - e-Grocer application
    - CF method
      - short term: below one percent
      - long-term, indirect effects important
- This study
  - measuring impact of different RS algorithms in Mobile Internet scenario
  - more than 3% more sales through personalized item ordering

The application context

- Game download platform of telco provider
  - access via mobile phone
  - direct download, charged to monthly statement
  - low cost items (0.99 cent to few Euro)
- Extension to existing platform
  - "My recommendations"
  - in-category personalization (where applicable)
  - start-page items, post-sales items
- Control group
  - natural or editorial item ranking
  - no "My Recommendations"

Study setup (A/B test)

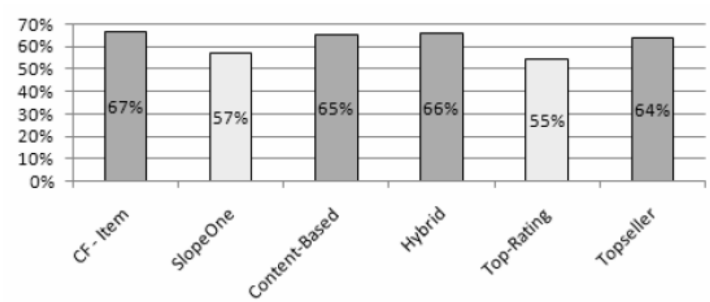
- 6 recommendation algorithms, 1 control group
  - CF (item-item, SlopeOne), Content-based filtering, Switching CF/Content-based hybrid, top rating, top selling
- Test period:
  - 4 weeks evaluation period
  - about 150,000 users assigned randomly to different groups
  - only experienced users
- Hypotheses on personalized vs. non-personalized recommendation techniques and their potential to
  - Increase conversion rate (i.e. the share of users who become buyers)
  - Stimulate additional purchases (i.e. increase the average shopping basket size)

Measurements

- Click and purchase behavior of customers
  - Customers are always logged in
  - All navigation activities stored in system
- Measurements taken in different situations
  - "My Recommendations", start page, post sales, in categories, overall effects
  - Metrics
    - item viewers/platform visitors
    - item purchasers/platform visitors
    - item views per visitor
    - purchases per visitor
- Implicit and explicit feedback
  - item view, item purchase, explicit ratings

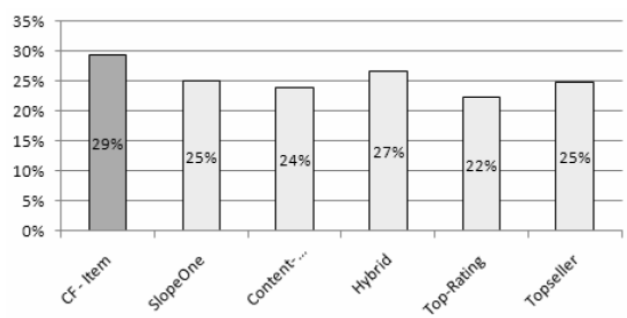
"My Recommendations" conversion rates

- Conversion rates
  - Top-rated items (SlopeOne, Top-Rating) appear to be non-interesting



viewers/visitors

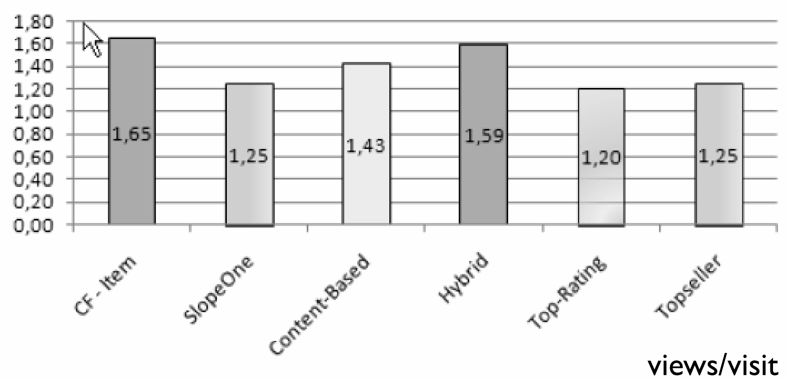
  - Only CF-Item able to turn more visitors into buyers ($p < 0.01$)



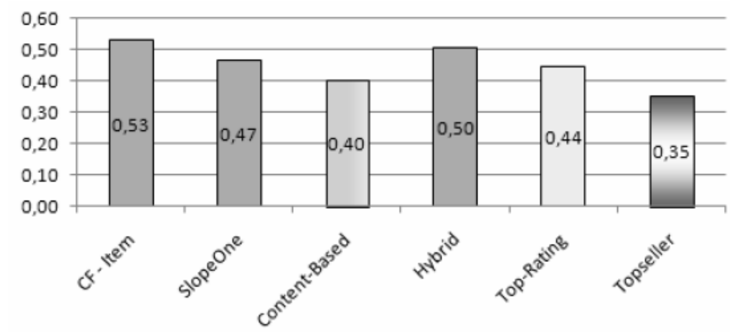
buyers/visitors

- Overall on the platform
  - No significant increase on both conversion rates (for frequent users!)

"My Recommendations" sales increase (1)



views/visit



purchases/visit

- Item views:
  - Except SlopeOne, all personalized RS outperform non-personalized techniques
- Item purchases
  - RS measurably stimulate users to buy/download more items
  - Content-based method does not work well here

"My Recommendations" sales increase (2)

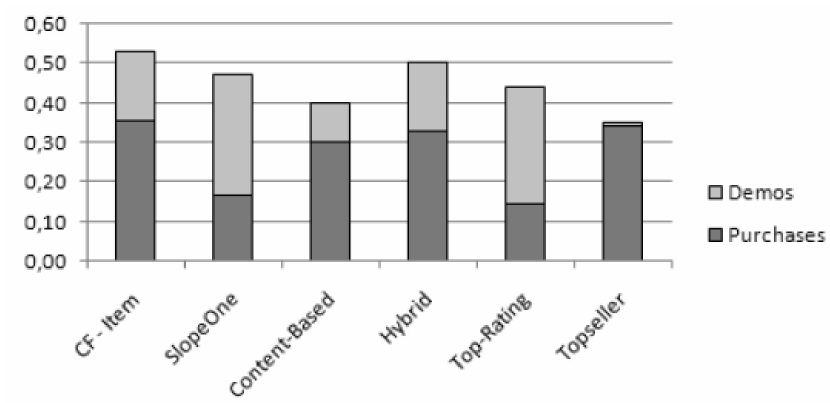
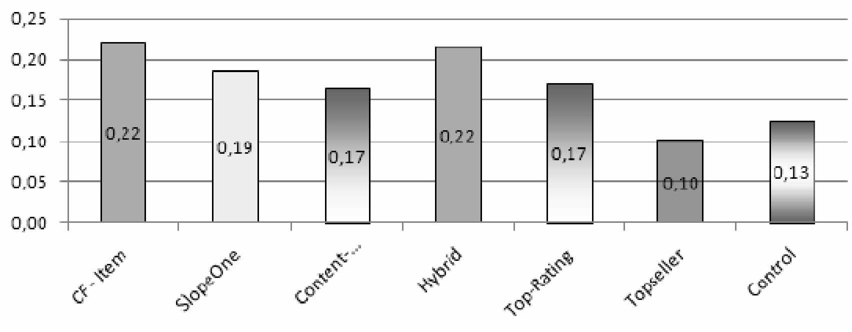
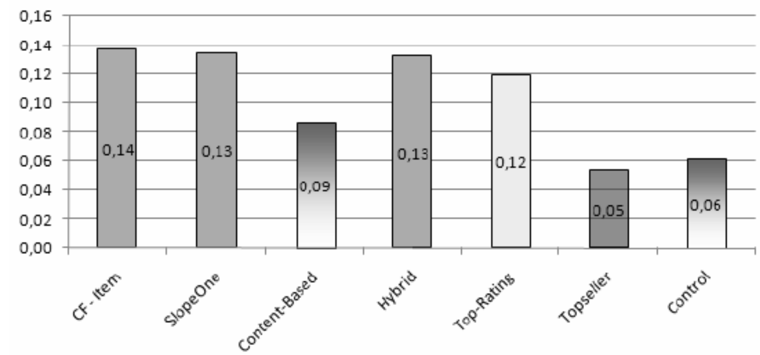


Figure shows purchases per visitor rate

- Demos and non-free games:
  - Previous figures counted all downloads
  - Figure shows
    - Personalized techniques comparable to top seller list
    - However, can stimulate interest in demo games

Post-sales recommendations

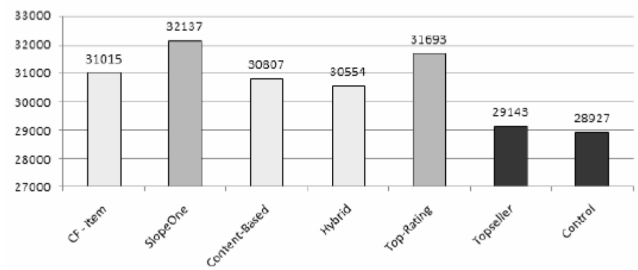




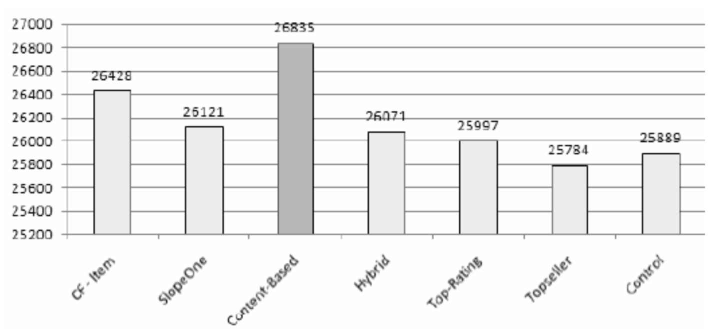
- Findings
  - recommending "more-of-the-same", top sellers or simply new items does not work well
  - Top-Rating and SlopeOne nearly exclusively stimulate demo downloads (Not shown)
  - Top-Seller und control group sell no demos

Overall effects

- Overall number of downloads
  - free + non-free games



- Pay games only



- Notes:
  - In-category measurements not shown in paper
  - Content-based method outperforms others in different categories
    - half price, new games, erotic games
  - Effect: 3.2 to 3.6% sales increase!

Observations & Summary

- Only 2% of users issued at least one rating
  - Most probably caused by size of displays
    - In addition: particularity of platform; rating only after download
  - Explicit feedback not sufficient, implicit feedback required
- Recommendation in navigational context
  - Acceptance of recommendation depends on situation of user
- Summary
  - Significant sales increase can be reached!
    - max. 1% in past with other activities
  - More studies needed
  - Limitations of accuracy measures

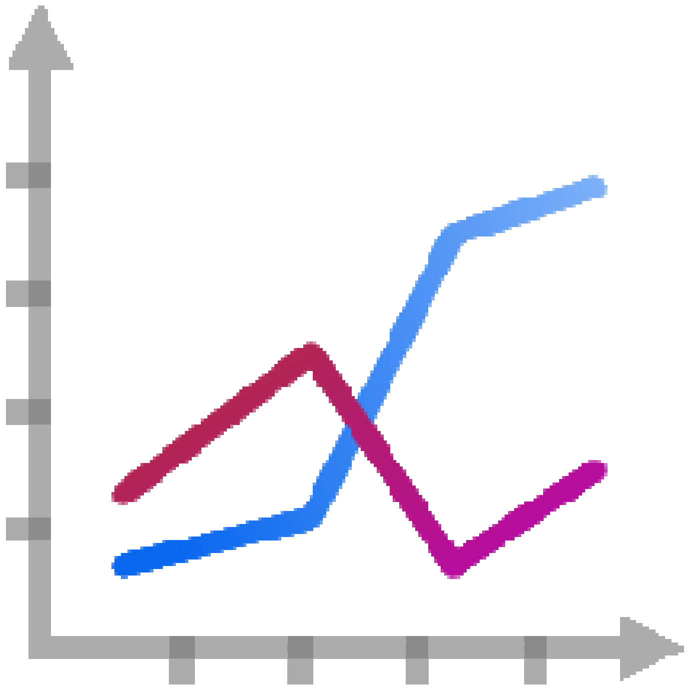
Recommender Systems
An introduction

Dietmar Jannach, TU Dortmund, Germany

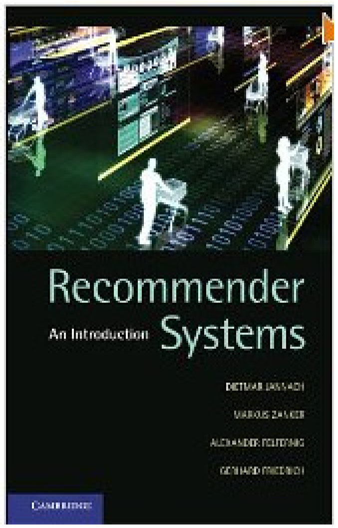
Slides presented at PhD School 2014, University Szeged, Hungary

dietmar.jannach@tu-dortmund.de

Evaluating recommender systems



What is a good recommendation?



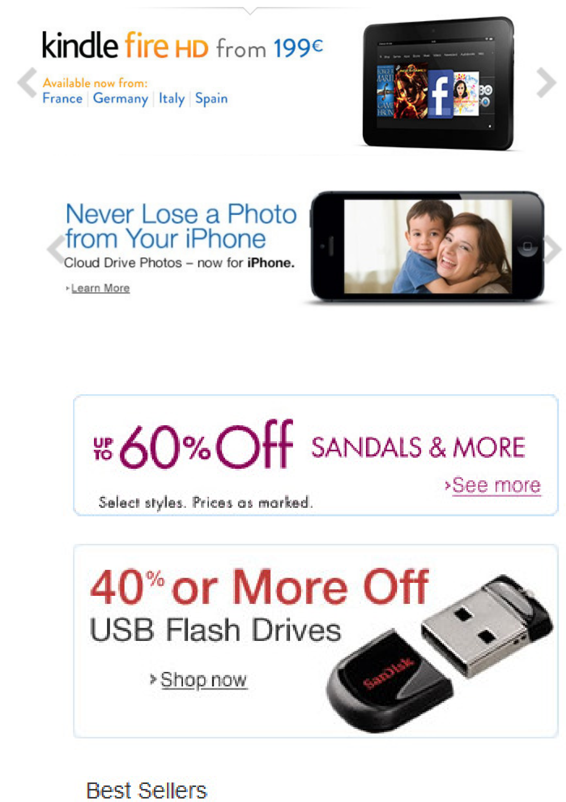
- This might lead to …
  - What is a good recommendation?
  - What is a good recommendation **strategy**?
  - What is a good recommendation strategy **for my business**?

These have been in stock for quite a while now …

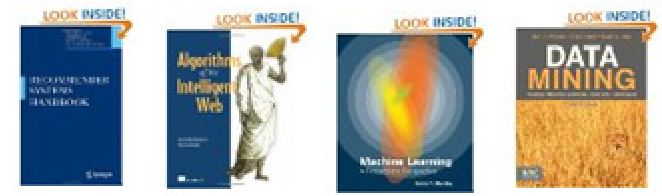
What is a good recommendation?

What are the measures in practice?

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

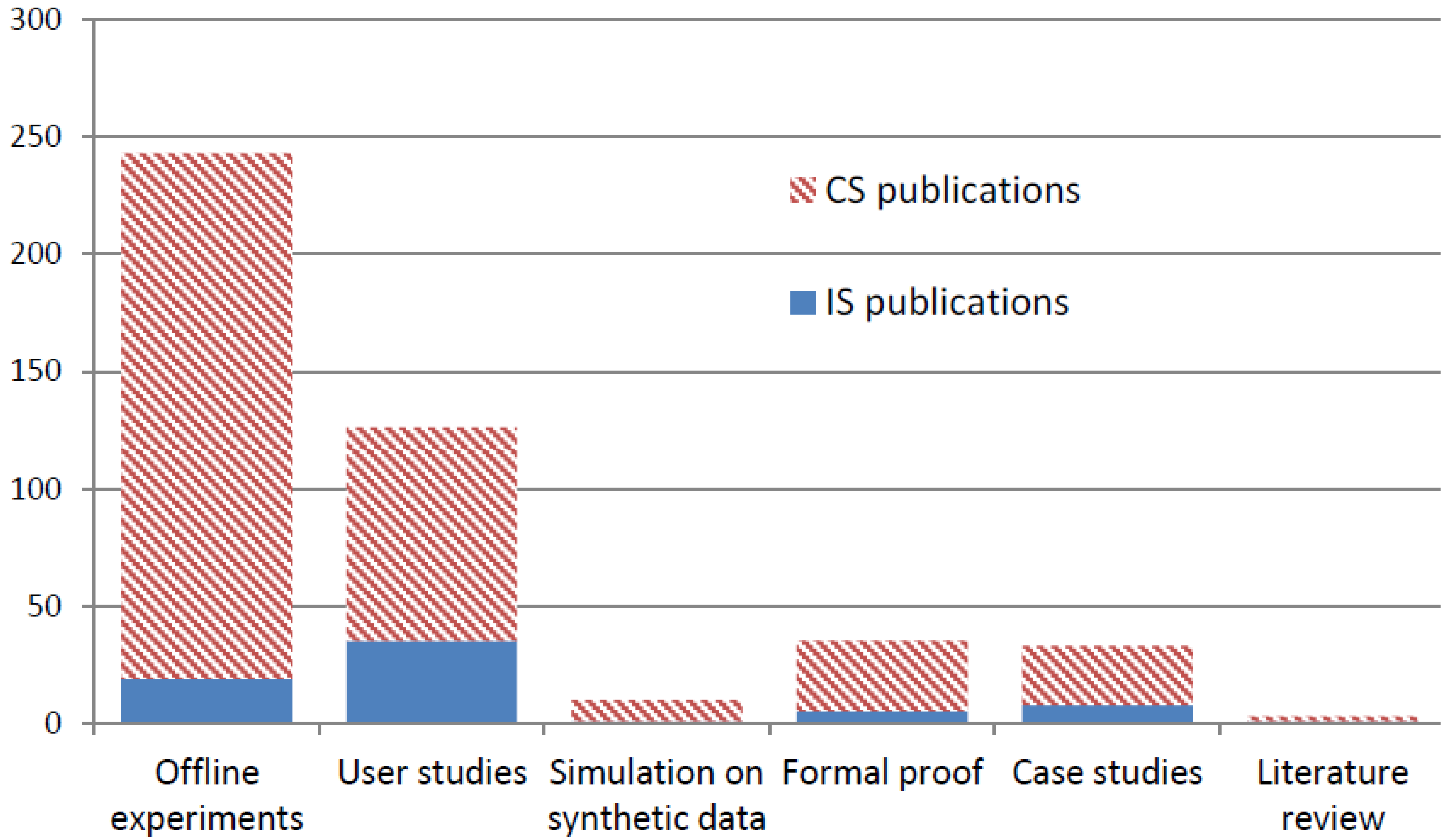


How do we as researchers know?



- Test with real users
  - A/B tests
  - Example measures: sales increase, click through rates
- Laboratory studies
  - Controlled experiments
  - Example measures: satisfaction with the system (questionnaires)
- Offline experiments
  - Based on historical data
  - Example measures: prediction accuracy, coverage

In academia – evaluation approaches



Empirical research

- Characterizing dimensions:
  - Who is the **subject** that is in the focus of research?
  - What **research methods** are applied?
  - In which **setting** does the research take place?

Subject	Online customers, students, historical online sessions, computers, …
Research method	Experiments, quasi-experiments, non-experimental research
Setting	Lab, real-world scenarios

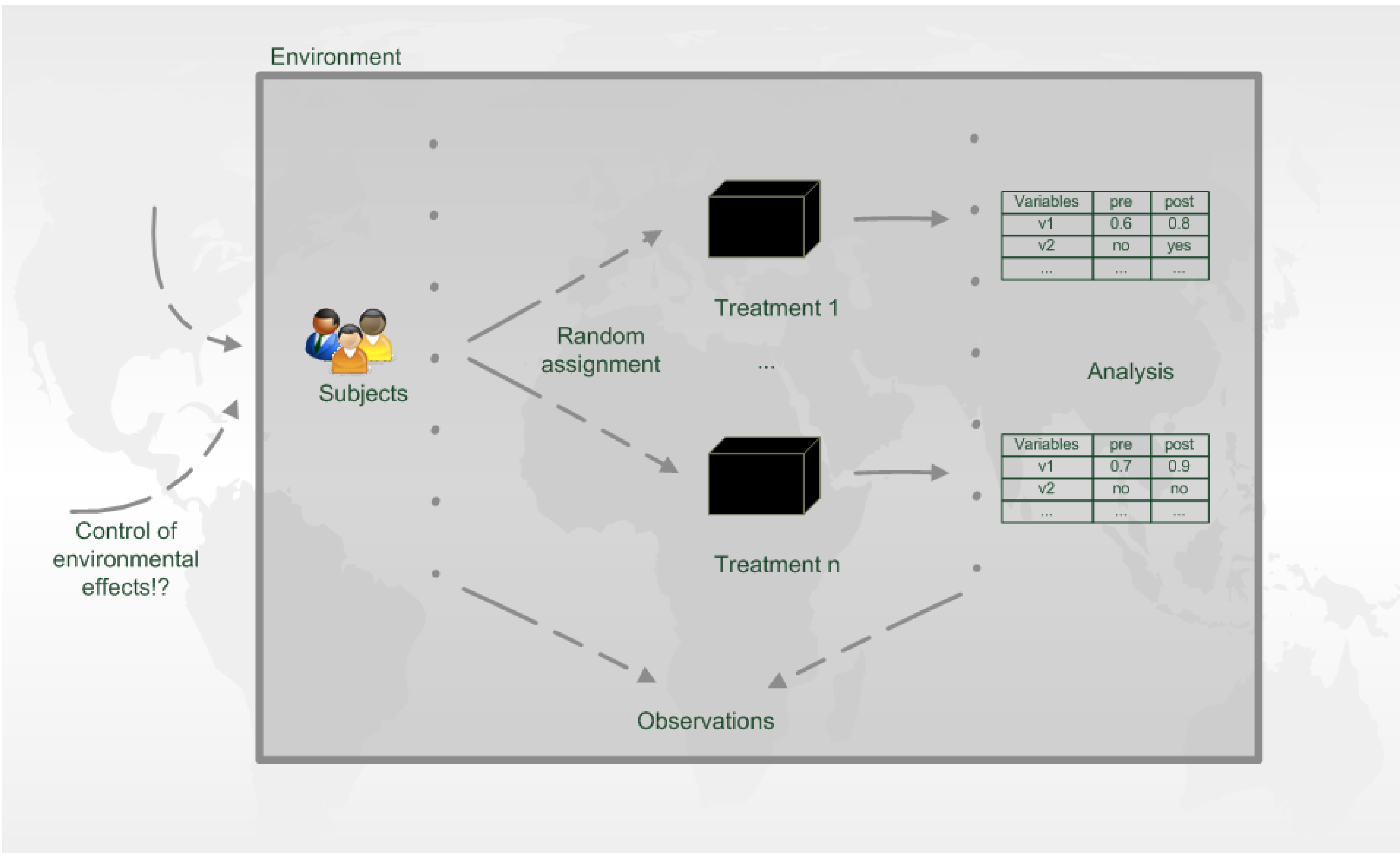
Evaluation settings (w. users)

- Lab studies
  - Explicitly created for the purpose of the study
  - Extraneous variables can be controlled more easy by selecting study participants
  - But doubts may exist about participants motivated by money or prizes
- Participants should behave as they would in a real-world environment
  - But they actually do not buy things
- Field studies
  - Conducted in an preexisting real-world environment
  - Users are intrinsically motivated to use a system

Research methods

- Experimental vs. non-experimental (observational) research methods
  - Experiment (test, trial):
    - *"An experiment is a study in which at least one variable is manipulated and units are randomly assigned to different levels or categories of manipulated variable(s)."*
    - Units : users, historic sessions, …
    - Manipulated variable : type of RS, groups of recommended items, explanation strategies …
    - Categories of manipulated variable(s): content-based RS, collaborative RS
  - Different experimental designs
    - Between subjects
    - Within subjects
    - Mixed designs

Experiment designs



Different approaches in different fields

- „How to“ from different perspectives:
  - Information Retrieval
  - Machine Learning
  - HCI and Decision Support

Evaluation in information retrieval (IR)

- Historical Cranfield collection (late 1950s)
  - 1,398 journal article abstracts
  - 225 queries
  - Exhaustive relevance judgements (over 300K)
- Ground truth established by human domain experts

		Reality	
		Actually Good	Actually Bad
Prediction	Rated Good	True Positive (tp)	False Positive (fp)
	Rated Bad	False Negative (fn)	True Negative (tn)

All recommended items

All good items

Metrics: Precision and Recall

- Recommendation is viewed as information retrieval task:
  - Retrieve (recommend) all items which are predicted to be "good".
  - Compare with "hidden" elements for which the ground truth is known
- **Precision:** a measure of exactness, determines the fraction of relevant items retrieved out of all items retrieved
  - E.g. the proportion of recommended movies that are actually good

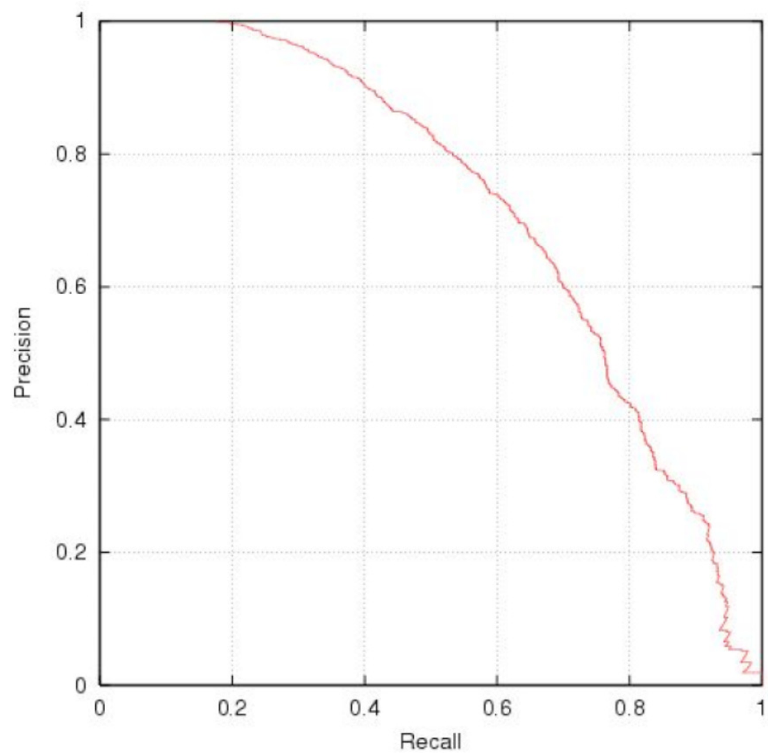
$$Precision = \frac{tp}{tp + fp} = \frac{|good\ movies\ recommended|}{|\text{all recommendations}|}$$

- **Recall:** a measure of completeness, determines the fraction of relevant items retrieved out of all relevant items
  - E.g. the proportion of all good movies recommended

$$Recall = \frac{tp}{tp + fn} = \frac{|good\ movies\ recommended|}{|all\ good\ movies|}$$

Precision vs. Recall

- E.g. typically when a recommender system is tuned to increase precision, recall decreases as a result (or vice versa)



F_1 Metric

- The **F_1 Metric** attempts to combine Precision and Recall into a single value for comparison purposes.
  - May be used to gain a more balanced view of performance

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

- The F_1 Metric gives equal weight to precision and recall
  - Other F_β metrics weight recall with a factor of β.

Precision@k, Recall@k, Mean Avg. Precision

- Precision@k/Recall@k
  - Define a threshold (list length) and count the "hits" proportion
- Mean Average Precision
  - Determine the position of each hit (e.g., 2,3,5)
  - Calculate the average for all hits in the list
  - Average over all recommendations
- Mean Reciprocal Rank
  - Assume that there is only one relevant item or only the first is important
  - If its position is K, the MRR is 1/K

Average Precision

- **Average Precision** (*AP*) is a ranked precision metric that places emphasis on highly ranked correct predictions (hits)
- Essentially it is the average of precision values determined after each successful prediction, i.e.

Rank	Hit?
1	
2	X
3	X
4	X
5	

$$AP = \frac{1}{3}\left(\frac{1}{2}+\frac{2}{3}+\frac{3}{4}\right) = \frac{23}{36} \approx 0.639$$

$$AP = \frac{1}{3}\left(\frac{1}{1}+\frac{2}{4}+\frac{3}{5}\right) = \frac{21}{30} = 0.7$$

Rank	Hit?
1	X
2	
3	
4	X
5	X

Metrics: Rank position matters

For a user:

Actually good
Item 237
Item 899

hit

Recommended (predicted as good)
Item 345
Item 237
Item 187

- **Rank metrics** extend recall and precision to take the positions of correct items in a ranked list into account
  - Relevant items are more useful when they appear earlier in the recommendation list
  - Particularly important in recommender systems as lower ranked items may be overlooked by users
- nDCG, Lift index, Rank Score

Discounted Cumulative Gain (DCG)

- Concept of graded relevance
  - Hits at the beginning count more (more "gain")
  - Documents of higher relevance are more important
  - Discounted gain at later positions
    - Often an exponential decay (half life) is assumed
      - e.g., based on the log function
  - Given a rank position p, and the graded relevance "rel" of an item I

$$\mathrm{DCG_p} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2(i)}$$

  - nDCG: Normalized value at length n
    - Compare with "ideal" ranking

nDCG example

- There are 6 items to rank: I1 to I6
- Relevance scores (0-3) scale:
  - 3,2,3,0,1,2
- DCG at 6:

$$\mathrm{DCG_6} = rel_1 + \sum_{i=2}^{6} \frac{rel_i}{\log_2 i} = 3 + (2 + 1.892 + 0 + 0.431 + 0.774) = 8.10$$

- An ideal ordering IDCG:
  - 3,3,2,2,1,0 would lead to an DCG of 8.69
- The nDCG
  - DCG/IDCG = 8.10/8.69 = 0.932

Wikipedia.org

Problem of the ground truth

- Often in Information Retrieval settings
  - Set of target documents is labeled with ground truth
- In recommender systems:
  - No rating available for most of the items
  - Considering unrated items as irrelevant?
  - Different ways of computing precision / recall
    - How to count the ranked elements with unknown ground truth

Task 1:
Rank algorithms using precision and recall

- How do you measure precision?
- How "wins" for Precision@3?

	Recommender A	Recommender B	Recommender C
Position	Ground truth	Ground truth	Ground truth
1	5	4	5
2	5	4	4
3	1	4	3
4	5	1	1
5	3	1	1

Task 1:
Rank algorithms using precision and recall

- And now?

	Recommender A	Recommender B	Recommender C
Position	Ground truth	Ground truth	Ground truth
1	?	4	?
2	5	?	4
3	?	4	3
4	5	?	1
5	3	1	?

Error measures – The Machine Learning perspective

- 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.
- Experimental setup
  - Historic user ratings constitute ground truth (e.g., MovieLens movie ratings, 100k ratings to 10 million; 100 mio. ratings for Netflix Prize)
  - Predict hidden ratings
  - **Mean Absolute Error** (*MAE*) computes the deviation between predicted ratings and actual ratings

$$MAE = \frac{1}{n}\sum_{i=1}^{n} | p_i - r_i |$$

  - **Root Mean Square Error** (*RMSE*) is similar to *MAE*, but places more emphasis on larger deviation

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (p_i - r_i)^2}$$

Example

Nr.	UserID	MovieID	Rating (r_i)	Prediction (p_i)	$\|p_i-r_i\|$	$(p_i-r_i)^2$
1	1	134	5	4.5	0.5	0.25
2	1	238	4	5	1	1
3	1	312	5	5	0	0
4	2	134	3	5	2	4
5	2	767	5	4.5	0.5	0.25
6	3	68	4	4.1	0.1	0.01
7	3	212	4	3.9	0.1	0.01
8	3	238	3	3	0	0
9	4	68	4	4.2	0.2	0.04
10	4	112	5	4.8	0.2	0.04
					4.6	5.6

- MAE = 0.46
- RMSE = 0.75

Removing line nr. 4

- MAE = 0.29
- RMSE = 0.42

Removing lines 1,2,4,5

- MAE = 0.1
- RMSE = 0.13

Dataset characteristics

- Natural datasets include historical interaction records of real users
  - Explicit user ratings
  - Datasets extracted from web server logs (implicit user feedback)
- Sparsity of a dataset is derived from ratio of empty and total entries in the user-item matrix:
  - Sparsity = $1 - |R|/(|I| \cdot |U|)$
  - R = ratings
  - I = items
  - U = users

The Netflix Prize setup

- Netflix competition
  - Web-based movie rental and streaming
  - Prize of $1,000,000 for accuracy improvement (RMSE) of 10% compared to own Cinematch system.
- Historical dataset
  - ~480K users rated ~18K movies on a scale of 1 to 5
  - ~100M ratings
  - Last 9 ratings/user withheld
    - Probe set – for teams for evaluation
    - Quiz set – evaluates teams' submissions for leaderboard
    - Test set – used by Netflix to determine winner

General methodology

- Setting to ensure internal validity:
  - One randomly selected share of known ratings (**training set**) used as input to train the algorithm and build the model
  - Model allows the system to compute recommendations at runtime
  - Remaining share of withheld ratings (**testing set**) required as ground truth to evaluate the model's quality
  - To ensure the reliability of measurements the random split, model building and evaluation steps are repeated several times
- N-fold cross validation is a stratified random selection procedure
  - N disjunct fractions of known ratings with equal size (1/N) are determined
  - N repetitions of the model building and evaluation steps, where each fraction is used exactly once as a testing set while the other fractions are used for training
  - Setting N to 5 or 10 is popular

Analysis of results

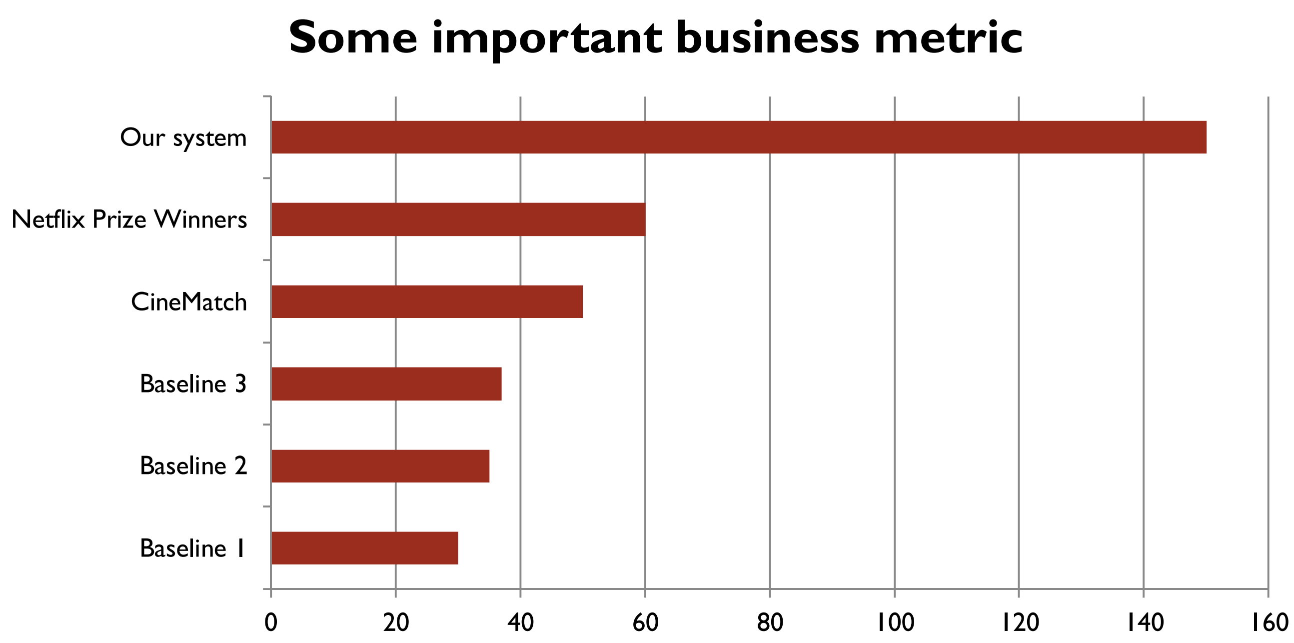
- Are observed differences statistically meaningful or due to chance?
  - Standard procedure for testing the statistical significance of two deviating metrics is the pairwise analysis of variance (ANOVA)
  - Null hypothesis H_0: observed differences have been due to chance
  - If outcome of test statistics rejects H_0, significance of findings can be reported
- Practical importance of differences?
  - Size of the effect and its practical impact
  - External validity or generalizability of the observed effects
  - Despite similar error metrics, algorithms can compare different sets of items
    - e.g., mostly popular, the same set to everyone

Reality check regarding F_1 and accuracy measures for RS

- Real value lies in increasing conversions
  - ... and satisfaction with bought items, low churn rate
- Some reasons why it might be a fallacy to think F_1 on historical data is a good estimate for real conversion:
  - Recommendation can be self-fulfilling prophecy
    - Users' preferences are not invariant, but can be constructed
  - Position/Rank is what counts (e.g. serial position effects)
    - Actual choices are heavily biased by the item's position
  - Smaller recommendation sets increase users' confidence in decision making
    - Effect of choice overload - large sets at the same time increase choice difficulty and reduce choice satisfaction
  - Inclusion of weak (dominated) items increases users' confidence
    - Replacing some recommended items by *decoy* items fosters choice towards the remaining options

Real-world check

- Presented at RecSys 2010
  - Research/Engineering Director, Netflix
  - Not the true numbers of course



Beyond accuracy – more quality metrics for recommenders

- Coverage
  - For how many users can we make recommendations?
  - How many catalog items are ever recommended?
- Diversity & Novelty
  - Avoiding monotone lists, discover new (families of) items
- Serendpity
  - Unexpected and surprising items might be valuable
- Familiarity
  - Give the user the impression of understanding his/her needs
- Biases
  - Does the recommender only recommend popular items and blockbusters?
- ….

Online experimentation

- Online study
  - Effectiveness of different algorithms for recommending cell phone games
  - Involved 150,000 users on a commercial mobile internet portal
  - Comparison of recommender methods in A/B tests
  - Random assignment of users to a specific method
  - Observation of customer behaviour
    - Increased number of item views / purchases
    - Increased conversion rates

A good recommendation?

Baseballschläger Aluminium von Tysonz

von Tysonz

(76 Kundenrezensionen)

Preis: EUR 15,00 - EUR 21,90

Alle Preisangaben inkl. MwSt.

Farbe: **Alu**

Größe:

Auswählen

- Aluminium Baseballschläger
- hochwertiges Aluminium
- robust & langlebig
- gummierter & rutschfester Griff
- mit TYSONZ Logo

Customers Who Bought This Item Also Bought

Teleskopschlagstock Abdrängstock 53 cm mit Moosgummigriff + Holster (63)
EUR 13,90

1 Paar Security Quarzsandhandschuhe aus echtem Leder (12)
EUR 15,95

KO Pfefferspray mit Sprühstrahl 40ml (204)
EUR 5,95
(EUR 148,75 / l)

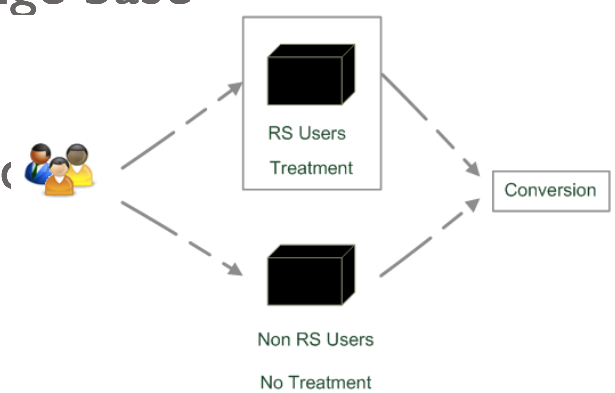
Wilson Baseball A1030 9 Inch (5)
EUR 11,55

VODOO Afrika Machete Jagdmachete XXL MESSER + Messerschärfer (18)

Security Handschuhe Quarzsandhandschuhe Defender mit Sandfüllung L (1)
EUR 19,95

Quasi-experimental settings

- SkiMatcher Resort Finder
  - introduced by Ski-Europe.com to provide users with recommendations based on their preferences
- Conversational RS
  - question and answer dialog
  - matching of user preferences with knowledge base
- Evaluation
  - Effectiveness of the recommender observed over a 4 month period in 2001
    - Classified as a quasi-experiment as users decide for themselves if they want to use the recommender or not



SkiMatcher Results

	July	August	September	October
Unique Visitors	10,714	15,560	18,317	24,416
• SkiMatcher Users	1,027	1,673	1,878	2,558
• Non-SkiMatcher Users	9,687	13,887	16,439	21,858
Requests for Proposals	272	506	445	641
• SkiMatcher Users	75	143	161	229
• Non-SkiMatcher Users	197	363	284	412
Conversion	2.54%	3.25%	2.43%	2.63%
• SkiMatcher Users	7.30%	8.55%	8.57%	8.95%
• Non-SkiMatcher Users	2.03%	2.61%	1.73%	1.88%
Increase in Conversion	359%	327%	496%	475%

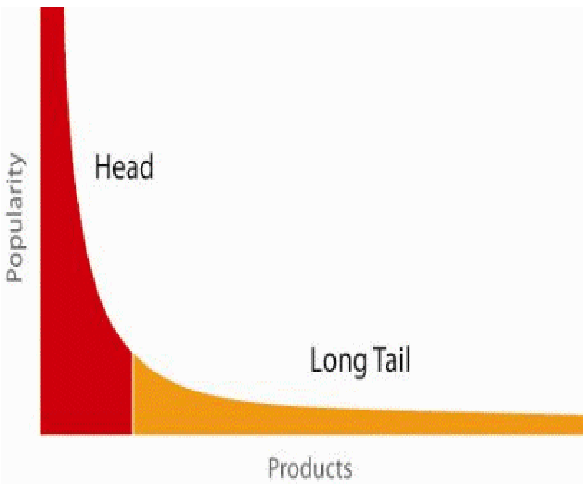
[Delgado and Davidson, ENTER 2002]

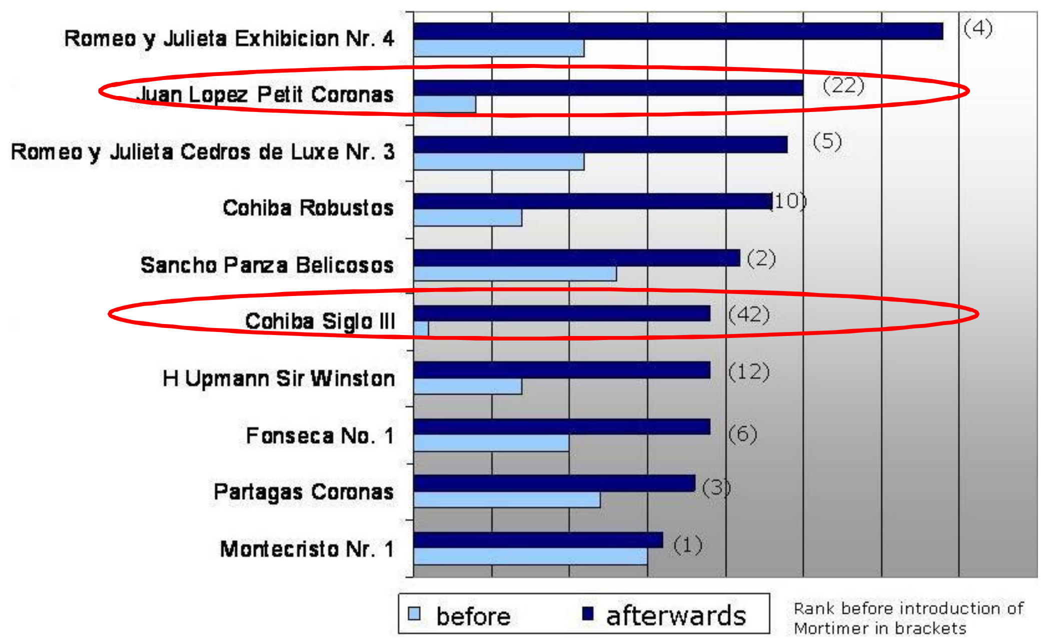
Interpreting the Results

- The nature of this research design means that questions of causality **cannot** be answered (lack of random assignments), such as
    - Are users of the recommender systems more likely convert?
    - Does the recommender system itself cause users to convert?
    - Some hidden exogenous variable might influence the choice of using RS as well as conversion.
- However, significant correlation between using the recommender system and making a request for a proposal
- Size of effect has been replicated in other domains
    - Tourism
    - Electronic consumer products

Observational research

- Increased demand in niches/long tail products
  - Books ranked above 250.000 represent >29% of sales at Amazon, approx. 2.3 million books [Brynjolfsson et al., Mgt. Science, 2003]
  - Ex-post from webshop data [Zanker et al., EC-Web, 2006]





Laboratory studies

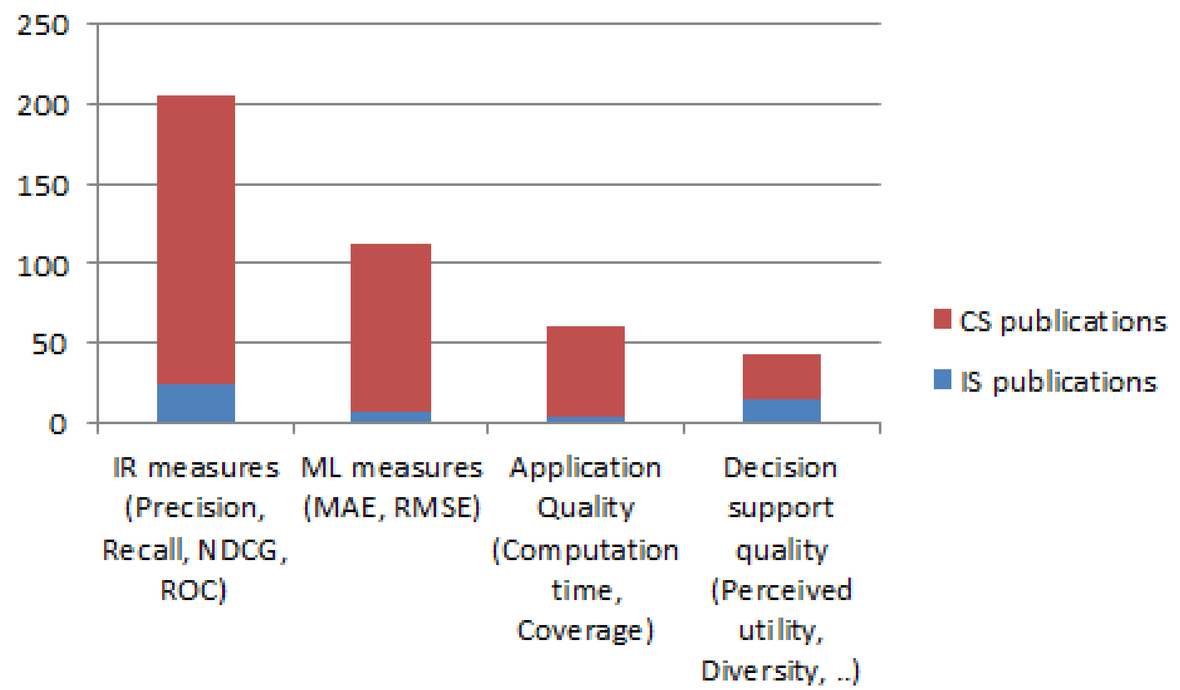
- Typical procedure
  - Develop hypothesis and design experimental setup
  - Develop two or more variants of a recommender system (treatments)
    - Variation can be in algorithm, presentation, user situation ..
  - Let participants use the system
    - between-subjects
      - Each participants "sees" one system
    - within-subjects (repeated measurements)
      - Participants uses all system
  - Measurements
    - Observations during the experiment (manual or automatic)
    - Questionnaire (before and) after the experiment
  - Analysis
    - Qualitative
    - Quantitative with statistical methods

Non-experimental research

- Quasi-experiments
  - Lack random assignments of units to different treatments
- Non-experimental / observational research
  - Surveys / Questionnaires
  - Longitudinal research
    - Observations over long period of time
    - E.g. customer life-time value, returning customers
  - Case studies
    - Focus on answering research questions about how and why
    - E.g., answer questions like: *How recommendation technology contributed to Amazon.com's becomes the world's largest book retailer?*
  - Focus group
    - Interviews
    - Think aloud protocols

Discussion & summary

- In RS, empirical evaluations on historical datasets dominates
  - Strong focus on accuracy measures
  - Limitations well known in the community



Toward multi-dimensional evaluation

- What is a good recommendation?
- Rating prediction is not enough
  - context matters, business goals can matter …
- Measures
  - Unclear if objective measures correspond to subjective experience
    - Reported differences are often tiny and probably dataset dependent
  - Probably domain-dependent
    - Content-based methods can work well in some domains
- Possibly desired characteristics of recommendation lists
  - diversity, novelty, serendipity, familiarity, homogeneity
  - Trade-off and multi-metric analysis required

Selection of own current work

- Looking into what recommenders recommend
  - largely different recommendations, even though
    - comparable accuracy results
    - from same family of algorithms

- How to deal with short-term preferences
  - Evaluation on real-world dataset
  - Short-term shopping goals are important

What recommenders recommend

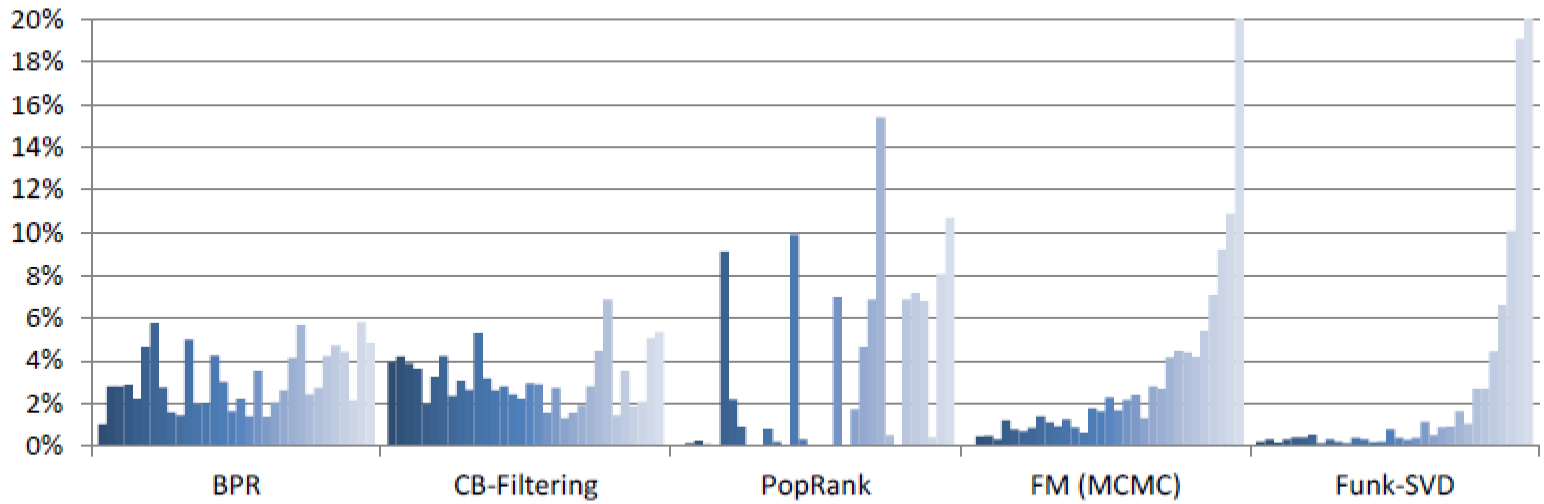
BPR	FUNK-SVD
Ace Ventura (1994)	**Shawshank Redemption (1994)**
Mrs. Doubtfire (1993)	Christmas Vacation (1989)
Pretty Woman (1990)	The World's Fastest Indian (2005)
The Mask (1994)	Life Is Beautiful (1997)
Beauty and the Beast (1991)	The Sixth Sense (1999)
Forrest Gump (1994)	Indiana Jones (Raiders) (1981)
Batman Forever (1995)	**Forrest Gump (1994)**
Independence Day (1996)	Indiana Jones (Crusade) (1989)
Waterworld (1995)	**The Dark Knight (2008)**
True Lies (1994)	Pirates of the Caribbean (2003)

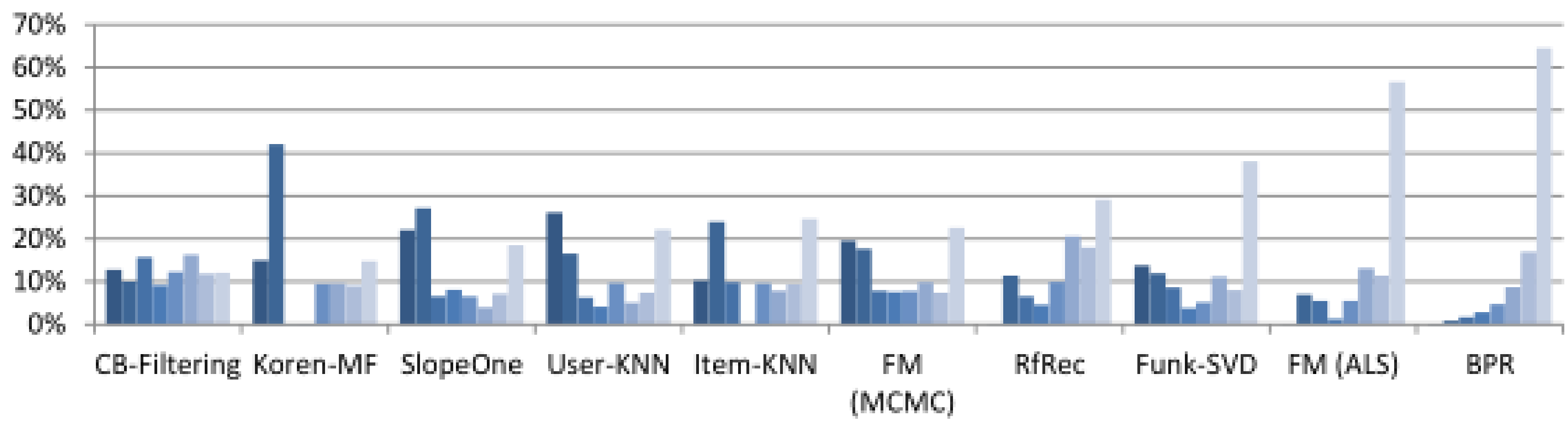
KOREN-MF	RF-REC
Shawshank Redemption (1994)	**Shawshank Redemption (1994)**
The Lives of Others (2006)	The Godfather (1972)
Paths of Glory (1957)	**The Lives of Others (2006)**
The Celebration (1998)	**Schindler's List (1993)**
One Flew Over the Cuckoo's Nest (1975)	**Ikiru (1952)**
Amelie (2001)	**The Dark Knight (2008)**
Double Indemnity (1944)	**Paths of Glory (1957)**
City of God (2002)	**The Celebration (1998)**
Ikiru (1952)	The Usual Suspects (1995)
Schindler's List (1993)	Casablanca (1942)

Accuracy results

Algorithm	RMSE	P@10(TS)	R@10(TS)	P@10(All)	R@10(All)	nDCG
Funk-SVD	**0.809**	0.426	0.799	0.071	0.117	**0.874**
FM (ALS)	0.814	0.425	0.798	0.093	0.148	0.872
FM (MCMC)	0.846	0.415	0.784	0.035	0.051	0.857
SlopeOne	0.855	0.411	0.780	0.028	0.045	0.854
User-kNN	0.856	0.411	0.781	0.036	0.065	0.856
Koren-MF	0.861	0.407	0.777	0.023	0.041	0.848
Rf-Rec	0.862	0.407	0.776	0.039	0.072	0.848
Item-kNN	0.863	0.407	0.777	0.030	0.057	0.849
WeightedAvg.	0.893	0.407	0.776	0.030	0.058	0.848
ItemAvgP	0.925	0.407	0.777	0.030	0.058	0.849
BPR	–	0.367	0.722	**0.129**	**0.290**	0.794
PopRank	–	0.353	0.709	0.083	0.178	0.790
CB-Filtering	–	0.345	0.698	0.021	0.038	0.774

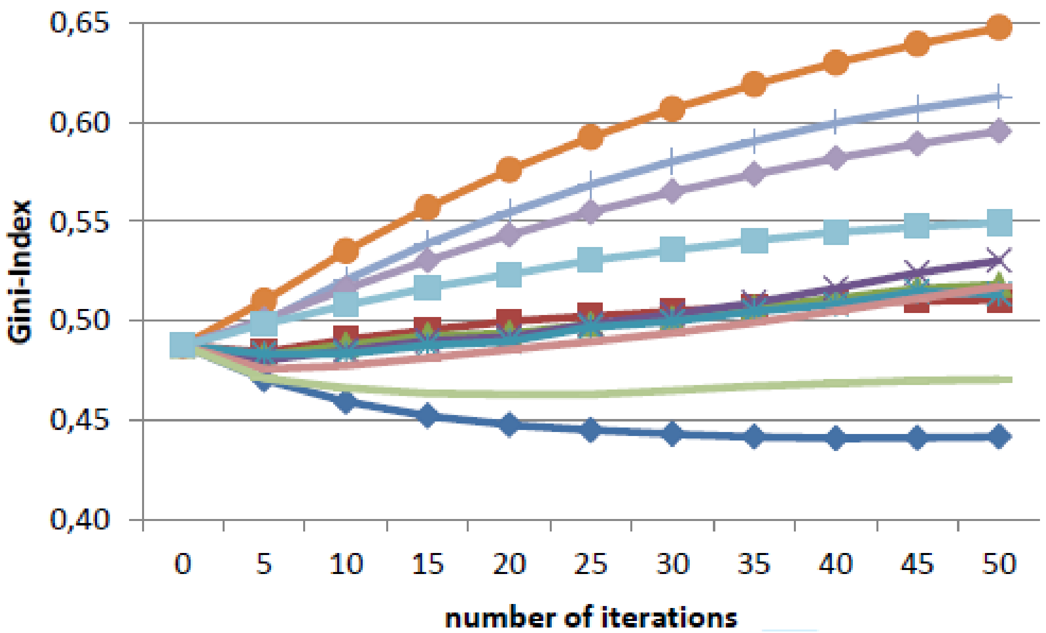
Distribution of recommendations by rating and popularity





Boosting blockbusters

- The rich become richer



Challenges in practical settings

- A cooperation project with Zalando
- What your returning customer has bought so far …

- Now she visits your shop and looks at this

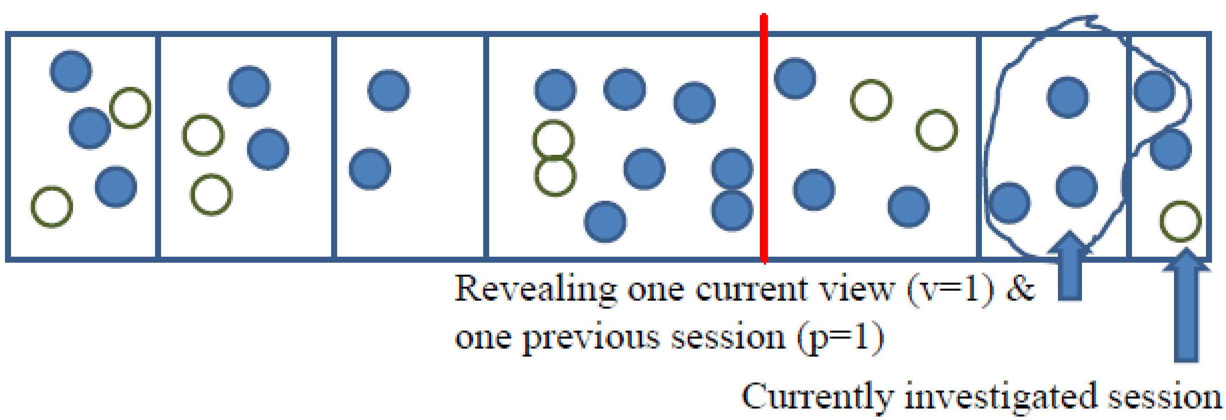
and then this

Challenges in practical settings

- Short-term preferences (shopping goals) are crucial
  - Can be considered as a sort of context
  - E.g., shopping for self or someone else?
- Adaptation to recent behavior must be immediate
  - No time to train or update complex models
- Long-term preferences can however be important
  - Preferred brands, colors, price segment, …
- Available information is huge and manifold – how to combine?
  - Sales, views, cart action, wish lists, search terms, category browsing
  - Billions of billions of data points (6 billion explicit ratings at Netflix)
  - Customer demographics
  - External factors like seasonal aspects, trends, time of the year …

Challenges in research

- Limitations of standard offline evaluation approaches
  - Typical:
    - Train-test cross validation with hidden ratings and accuracy metric
  - Here:
    - No ratings but various types of other information
    - Number of purchases and past interactions can be low
    - Time, and session-specific context and goals
- Proposal for alternative evaluation settings



Some experiments

- Different datasets created
- Various techniques compared
  - Popularity-based,
    BPR, item-item, co-occurrence, "feature matching" hybrid
  - Feature matching
    - Create a simple user profile based on item characteristics
      - brands, categories
    - Re-rank outputs of other technique
  - Recommend recently visited items

	Sparse	Medium	Dense
Users	53,328	27,137	12,760
Items	35,249	27,510	20,611
Purchases	437,848	292,912	182,817
Views	5,776,007	3,465,273	1,869,967
Min. purchases/user	3	7	10
Min. purchases/item	3	7	10

Results

- Protocol
  - Different levels of "revealing" context information
  - Current session and previous ones
  - Recall as a measurement
- Findings
  - Strong improvements possible despite simple strategies

	v=0, p=2	v=2, p=2	v=5, p=2	v=10, p=2	v=5, p=0
PopRank	0.14				
ContentPop	0.16				
BPR	0.57				
CoOccur	0.29	0.38	0.43	0.46	0.35
PopRank + FM	0.34	0.67	0.78	0.83	0.73
BPR+FM	0.64	0.77	0.84	**0.88**	0.82

Some open issues

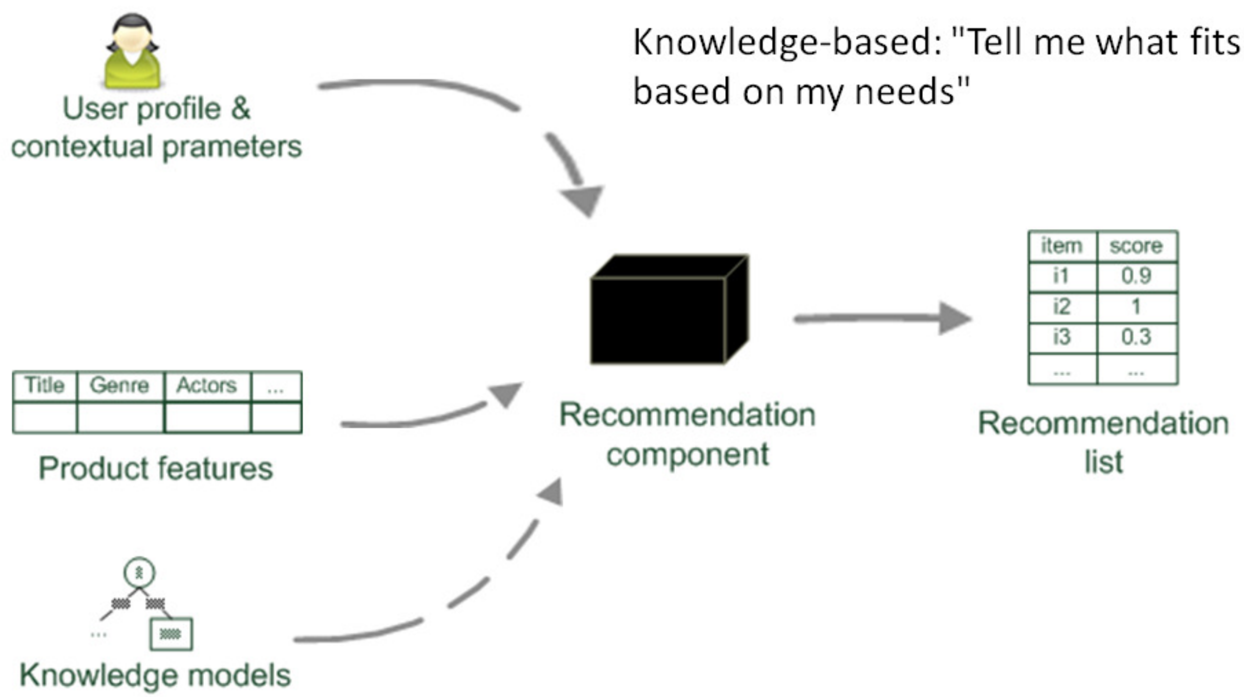
- How to interpret the user actions?
  - Views, Wishes, Carts, Purchases
- Should we recommend already seen items?
- Abundance of data
  - Every click is logged
    - Navigation and search actions could be relevant
  - Not all data available / shared
    - Specific item features might be relevant
- External factors not considered
  - Marketing campaigns
  - Seasonal aspects

Recommender Systems
An introduction

Dietmar Jannach, TU Dortmund, Germany
Slides presented at PhD School 2014, University Szeged, Hungary

dietmar.jannach@tu-dortmund.de

Knowledge-based approaches



Why do we need knowledge-based recommenders?

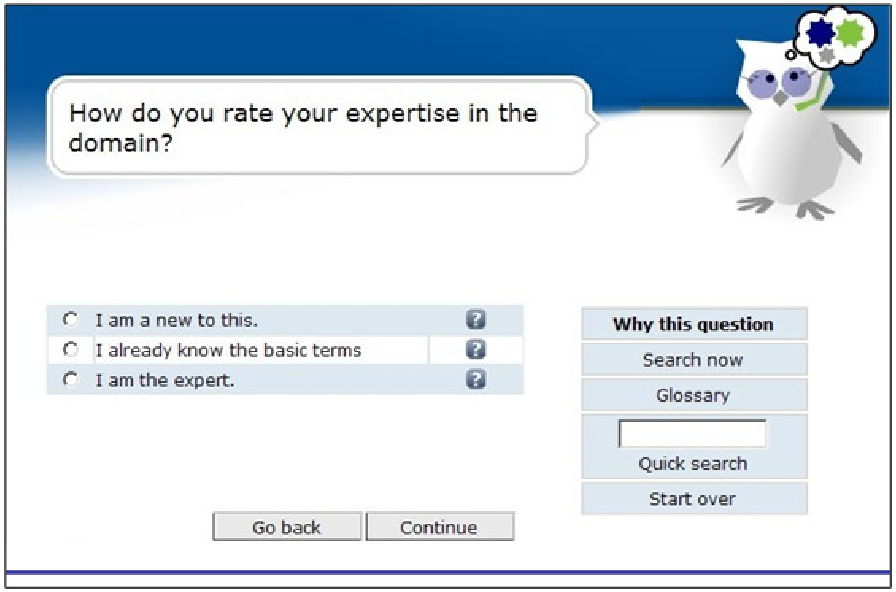
- Products with low number of available ratings

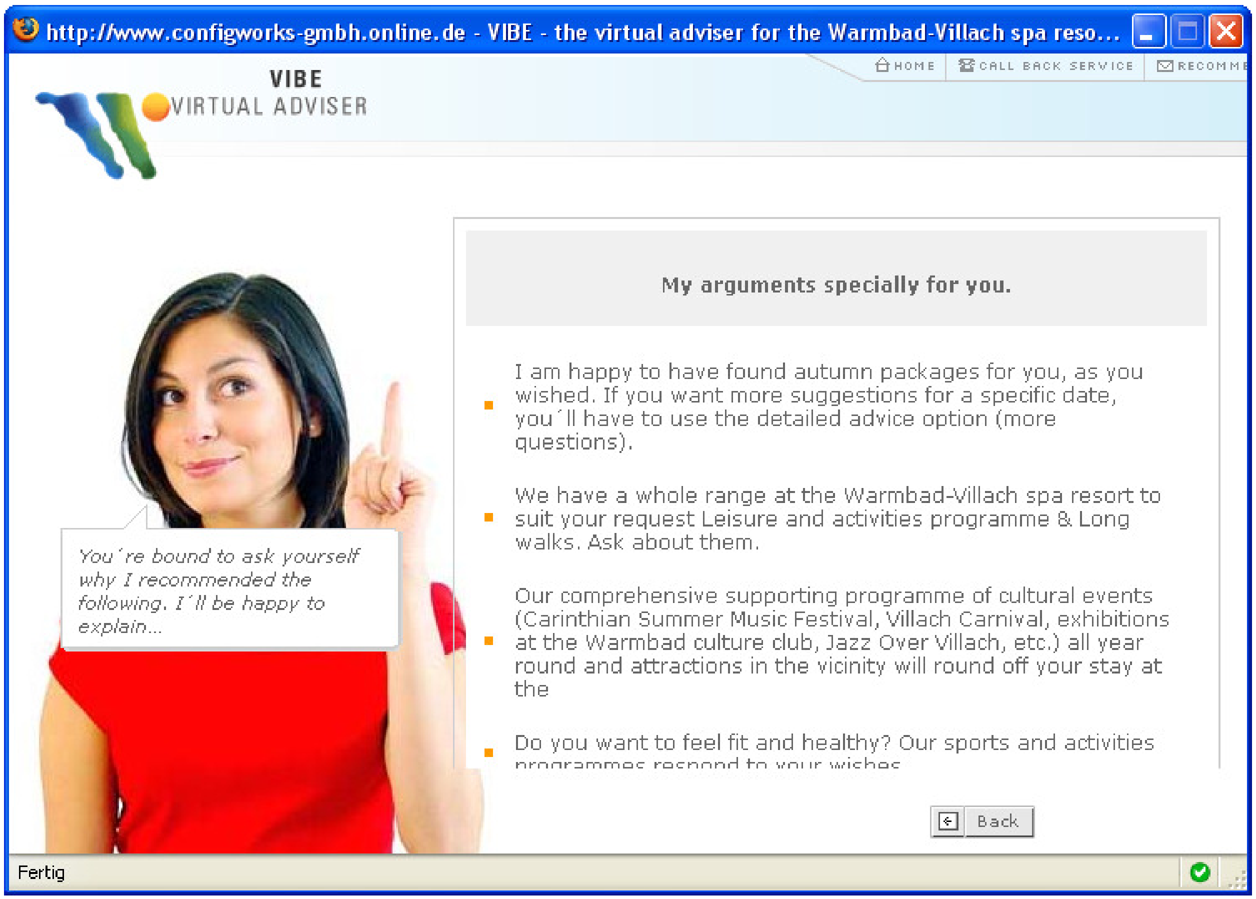
- Time span plays an important role
  - five-year-old ratings for computers
  - user lifestyle or family situation changes
- Customers want to define their requirements explicitly
  - "the color of the car should be black"

Knowledge-based (interactive) approaches

- Recommend items based on explicit knowledge
  - Acquire the user preferences interactively
    - e.g., through a series of web forms
  - Recommend items based on knowledge about how to match preferences with given item features
    - various types of matching approaches
  - Typical approaches
    - constraints, rules, similarities, utility functions, case-based reasoning



Example: An interactive travel recommender



- Personalized preference elicitation
- Customized buying proposal
- Explanation / argumentation

Knowledge-Based Recommendation

- Explicit domain knowledge
  - Sales knowledge elicitation from domain experts
  - System mimics the behavior of experienced sales assistant
  - Best-practice sales interactions
  - Can guarantee "correct" recommendations (determinism) with respect to expert knowledge

- Conversational interaction strategy
  - Opposed to one-shot interaction
  - Elicitation of user requirements
  - Transfer of product knowledge ("educating users")

Knowledge-Based Recommendation

- Different views on “knowledge”
  - Similarity functions
    - Determine matching degree between query and item (case-based RS)
  - Utility-based RS
    - E.g. MAUT – Multi-attribute utility theory
  - Logic-based knowledge descriptions (from domain expert)
    - E.g. Hard and soft constraints

- Hybridization
  - E.g. ‚merging explicit knowledge with community data
  - Can ensure some policies based on e.g. availability, user context or profit margin

Typical Approaches

- Constraint-based
  - based on explicitly defined set of recommendation rules (constraints)
  - retrieve items that fulfill recommendation rules and user requirements
- Case-based systems / critiquing
  - based on different types of similarity measures
  - retrieve items that are similar to user requirements
- Both approaches are similar in their conversational recommendation process
  - users specify the requirements
  - recommender system tries to identify solutions
  - if no solution can be found, users can change their requirements

Constraint-based Recommendation

- Knowledge base
  - connects user preferences (model) and item features
  - variables
    - user model features (requirements), item features (catalogue)
  - set of constraints
    - logical implications (IF user requires A THEN proposed item should possess feature B)
    - hard and soft/weighted constraints
    - solution preferences
- Derive a set of recommendable items
  - items fulfill requirements and constraints
  - explanations – transparent line of reasoning
    - why this recommendation?
    - why was no solution found and how to deal with this situation?

An example problem

- Select items from this catalog that match the user's requirements

id	price(€)	mpix	opt-zoom	LCD-size	movies	sound	waterproof
P_1	148	8.0	4×	2.5	no	no	yes
P_2	182	8.0	5×	2.7	yes	yes	no
P_3	189	8.0	10×	2.5	yes	yes	no
P_4	196	10.0	12×	2.7	yes	no	yes
P_5	151	7.1	3×	3.0	yes	yes	no
P_6	199	9.0	3×	3.0	yes	yes	no
P_7	259	10.0	3×	3.0	yes	yes	no
P_8	278	9.1	10×	3.0	yes	yes	yes

- User's requirements can, for example, be
  - "the price should be lower than 300 €"
  - "the camera should be suited for sports photography"

Finding a set of suitable items 1

- Rule-based filtering with conjunctive queries
  - Rules:
    - if user choses "low" price, recommend cameras with price < 300
    - if user choses "nature photography", recommend cameras with more than 10 mega pixels
  - Conjunctive queries
    - Create a conjunctive query ("and" expression) from the right hand side of the matching rules
    - Run against database
  - Easy implementation
  - In case no matching product remains
    - Possible compromises for the user can be efficiently calculated in memory

Finding a set of suitable items 1

- Rule-based filtering with conjunctive queries
  - Rules:
    - if user choses "low" price, recommend cameras with price < 300
    - if user choses "nature photography", recommend cameras with more than 10 mega pixels
  - Conjunctive queries
    - Create a conjunctive query ("and" expression) from the right hand side of the matching rules
    - Run against database
  - Easy implementation
  - In case no matching product remains
    - Possible compromises for the user can be efficiently calculated in memory

Finding a set of suitable items 2

- Encode the problem as Constraint Satisfaction Problem
- Constraint Satisfaction Problems (CSP)
  - Basically, a very simple model consisting of
    - Variables having a defined and typically finite domains
    - Constraints that describe allowed value assignments to the variables
  - The problem
    - Find an assignment of values to all variables, such that no constraint is violated
- Solution search
  - Problem is NP complete in general
  - Many practically relevant problems however tractable
  - Efficient solver implementations exist

Knowledge-based recommendation encoded as CSP

- The recommendation problem can be encoded as follows:

$$CSP(X_I \cup X_U, D, SRS \cup KB \cup I)$$

- Definitions
  - X_I, X_U: Variables describing product and user model with domain D
    - e.g., display size, optical zoom, price preference of user, purpose…
  - KB: Knowledge base with domain restrictions
    - e.g**. if** purpose=*on travel* **then** lower focal length < *28mm*
  - SRS: Specific requirements of user (e.g. purpose = *on travel*)
  - I: Product catalog
    - e.g. (id=1 ∧ lfl = 28mm) ∨ (id=2 ∧ lfl= 35mm) ∨ …)

- Solution: An assignment tuple θ assinging values to all variables X_I

$s.t.\ SRS \cup KB \cup I \cup \theta$ is satisfiable

Additional reasoning with knowledge-based approaches

- The explicit nature of the problem encoding allows various types of reasoning
  - What if the user's requirements cannot be fulfilled? What if they are user requirements are inconsistent?
    - Find a "relaxation" or "compromise"
  - What if the knowledge base is inconsistent?
    - Find a "diagnosis"
  - Why was a certain item (not?) recommended
    - Compute logical explanations

Reasoning - example

- What if no solution exists?

$KB \cup I$	not satisfiable	→ debugging of knowledge base
$SRS \cup KB \cup I$	not satisfiable but	
$KB \cup I$	satisfiable	→ debugging of user requirements

- Application of model-based diagnosis for debugging user requirements
  - Diagnoses: $(SRS \setminus \Delta) \cup KB \cup I$ is satisfiable
  - Repairs: $(SRS \setminus \Delta) \cup \Delta_{repair} \cup KB \cup I$ is satisfiable
  - Conflict sets: $CS \subseteq SRS : CS \cup KB \cup I$ is not satisfiable

Example: find minimal relaxations (minimal diagnoses)

Knowledge Base:

	LHS	RHS
C1	TRUE	Brand = Brand pref.
C2	Motives = *Landscape*	Low. foc. Length =< *28*
C3	TRUE	Price =< Max. cost

Product catalogue:

Powershot XY	
Brand	*Canon*
Lower focal length	*35*
Upper focal length	*140*
Price	*420 EUR*

Lumix	
Brand	*Panasonic*
Lower focal length	*28*
Upper focal length	*112*
Price	*319 EUR*

Current user:

		User model (SRS)	
CS1	R1	Motives	*Landscape*
	R2	Brand preference	*Canon*
CS2	R3	Max. cost	*350 EUR*

Diagnoses: $\Delta_1 = \{R2\}, \Delta_2 = \{R1, R3\}$

Ranking the items

- A CSP/conjunctive query encoding does not entail a ranking of the solution
- Possible approaches:
  - In case of unsatisfiable requirements
    - Rank those items highly that fulfill most constraitns
  - If there are many solutions
    - Use a distance function to determine the "closest" solution
    - Use a utility-model to rank the items
      - e.g., based on Multi-Attribute Utility Theory

Ranking with MAUT I

- Each item has several *quality dimensions*
  - Attribute values contribute to those dimensions
- *Quality and economy* could be *dimensions in* the domain of digital cameras

id	value	quality	economy
price	≤250	5	10
	>250	10	5
mpix	≤8	4	10
	>8	10	6
opt-zoom	≤9	6	9
	>9	10	6
LCD-size	≤2.7	6	10
	>2.7	9	5
movies	Yes	10	7
	no	3	10
sound	Yes	10	8
	no	7	10
waterproof	Yes	10	6
	no	8	10

Ranking with MAUT 2

- Consider the customer interest in these dimens
- Customer specific interest

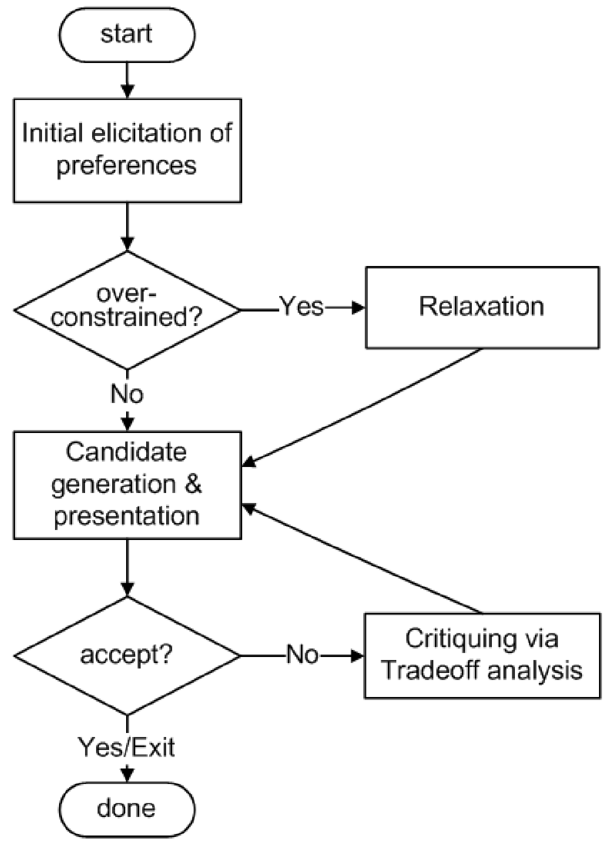
Customer	quality	economy
Cu_1	80%	20%
Cu_2	40%	60%

- Calculation of Utility

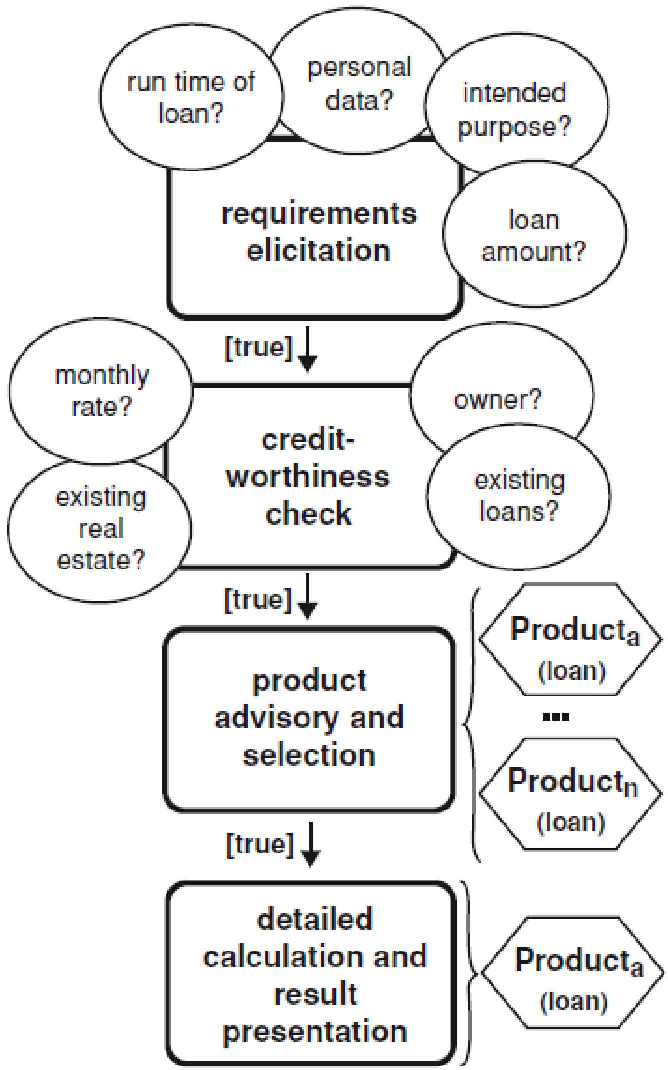
quality	economy	cu_1	cu_2
P_1 Σ(5,4,6,6,3,7,10) = 41	Σ (10,10,9,10,10,10,6) = 65	45.8 [8]	55.4 [6]
P_2 Σ(5,4,6,6,10,10,8) = 49	Σ (10,10,9,10,7,8,10) = 64	52.0 [7]	58.0 [1]
P_3 Σ(5,4,10,6,10,10,8) = 53	Σ (10,10,6,10,7,8,10) = 61	54.6 [5]	57.8 [2]
P_4 Σ(5,10,10,6,10,7,10) = 58	Σ (10,6,6,10,7,10,6) = 55	57.4 [4]	56.2 [4]
P_5 Σ(5,4,6,10,10,10,8) = 53	Σ (10,10,9,6,7,8,10) = 60	54.4 [6]	57.2 [3]
P_6 Σ(5,10,6,9,10,10,8) = 58	Σ (10,6,9,5,7,8,10) = 55	57.4 [3]	56.2 [5]
P_7 Σ(10,10,6,9,10,10,8) = 63	Σ (5,6,9,5,7,8,10) = 50	60.4 [2]	55.2 [7]
P_8 Σ(10,10,10,9,10,10,10) = 69	Σ (5,6,6,5,7,8,6) = 43	63.8 [1]	53.4 [8]

Interacting with constraint-based recommenders

- The user specifies his or her initial preferences
  - all at once or
  - incrementally in a wizard-style
  - interactive dialog
- The user is presented with a set of matching items
  - with explanation as to why a certain item was recommended
- The user might revise his or her requirements
  - see alternative solutions
  - narrow down the number of matching items

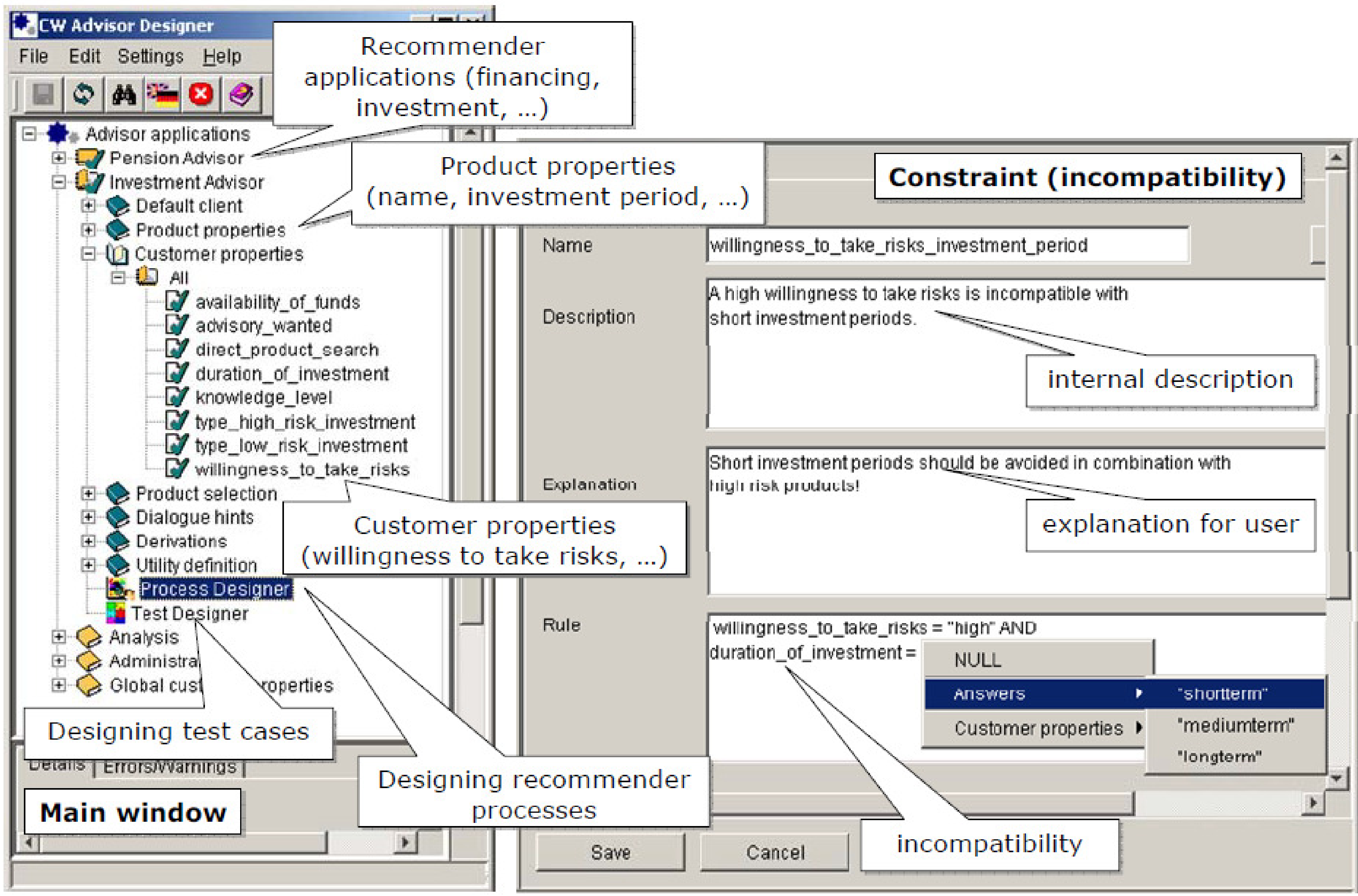


Example: sales dialogue financial services



- Complex multi-step preference and requirements elicitation
- Resembles call-center scripting
  - best-practice sales dialogues
  - Consistent quality
- Modeling support
  - States, transitions with predicates
  - Developed real-world deployed system
  - Comprehensive modeling environment

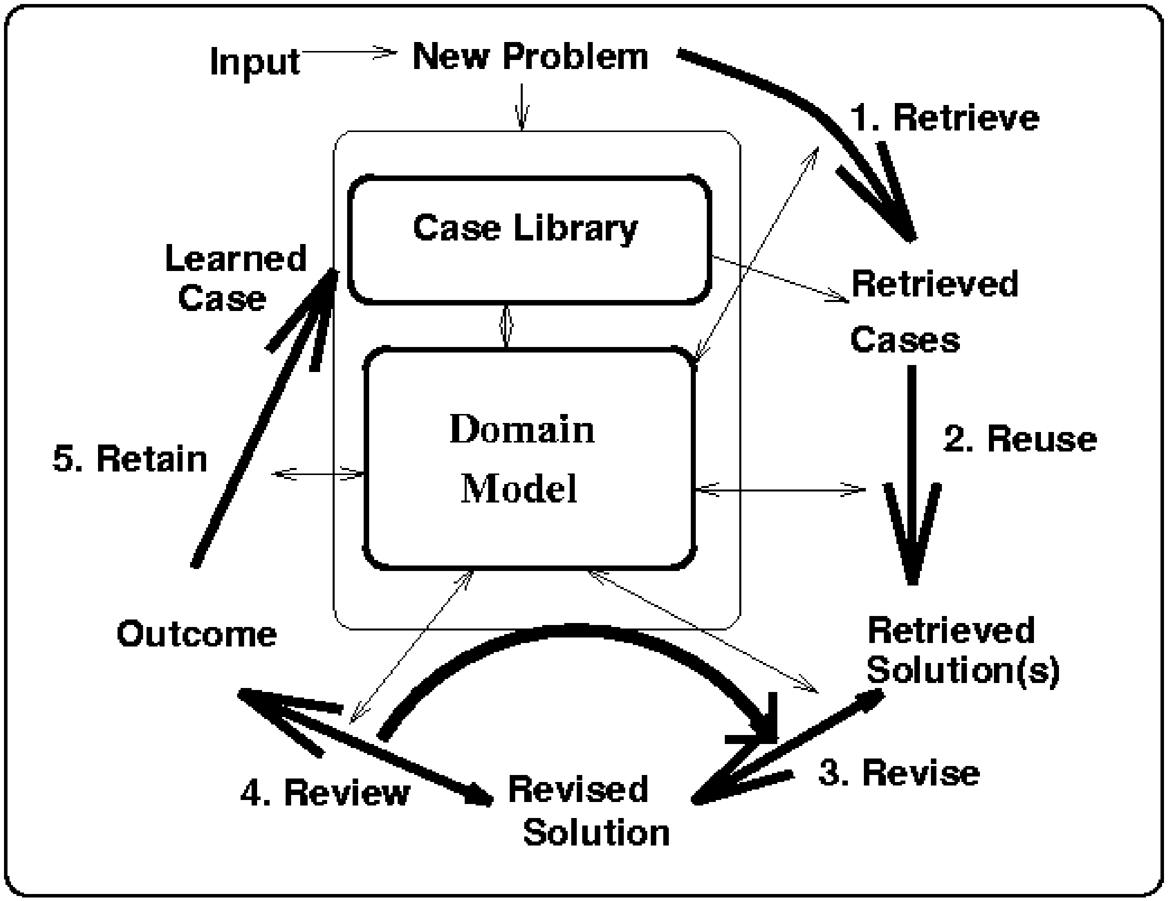
Example software: Advisor Suite



Case-based recommendation & Critiquing

- Idea of Case-based reasoning
  - "A case-based reasoner solves new problems by adapting solutions that were used to solve old problems"
  - CBR problem solving process:
    - Store previous experiences (cases) in memory
    - To solve new problems
      - Retrieve from the memory similar experience about similar situations
      - Reuse the experience in the context of the new situation: complete or partial reuse, or adapt according to differences
      - Store new experience in memory (learning)
  - Idea can be transferred to recommendation
    - However, not always clear what is still CBR and what not
    - Often, similarity functions are the main knowledge
    - "Critiquing" as an interaction style

Case-based reasoning

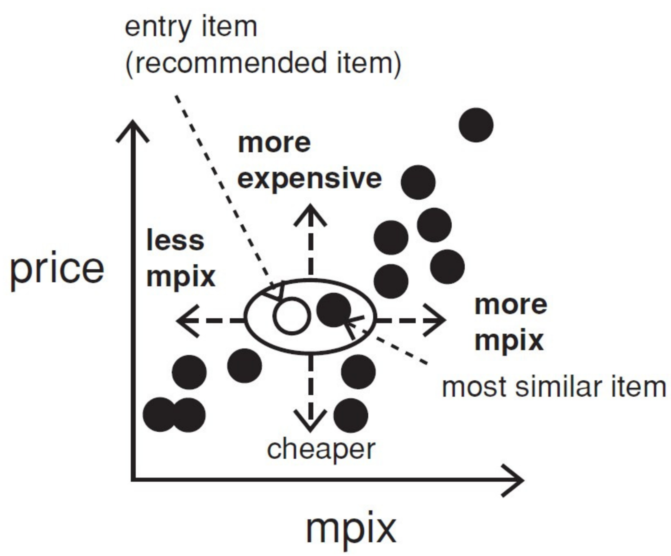


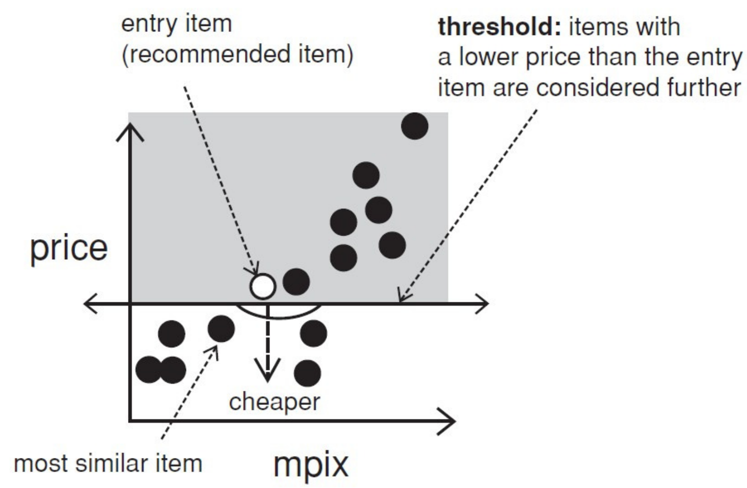
Critiquing

- Navigate the product space by "criticizing" the current solution
- Knowledge types:
  - About items
  - Adaptation step sizes
  - (Similarity functions)
- Example:
  - Looking for a restaurant …

The case for critiquing

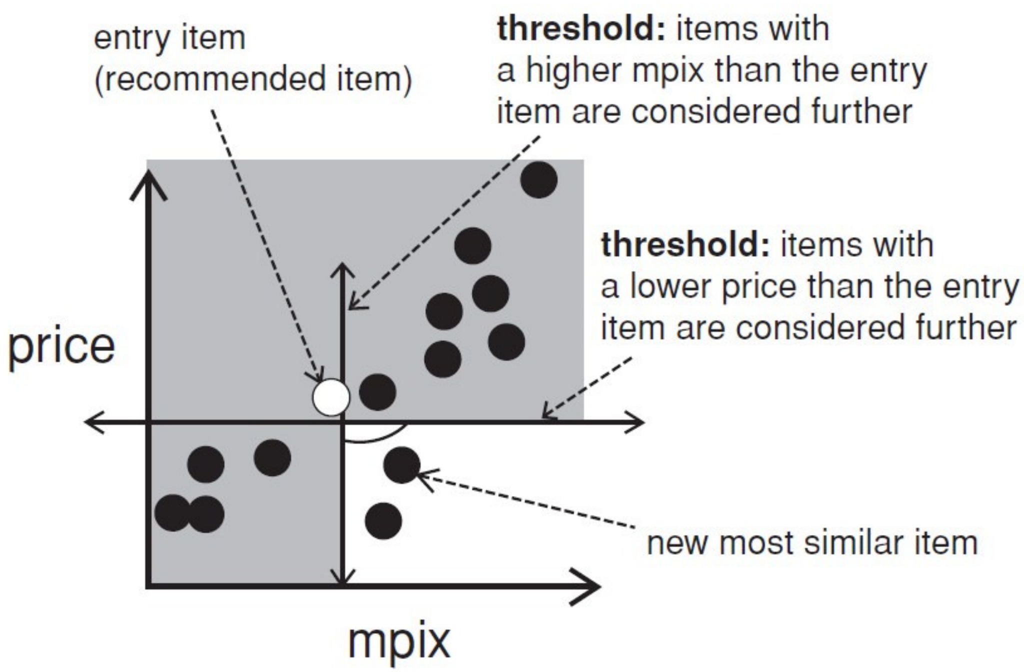
- Customers maybe not know what they are seeking
- Critiquing is an effective way to support such navigations
- Customers specify their change requests (price or mpix) that are not satisfied by the current item (entry item)



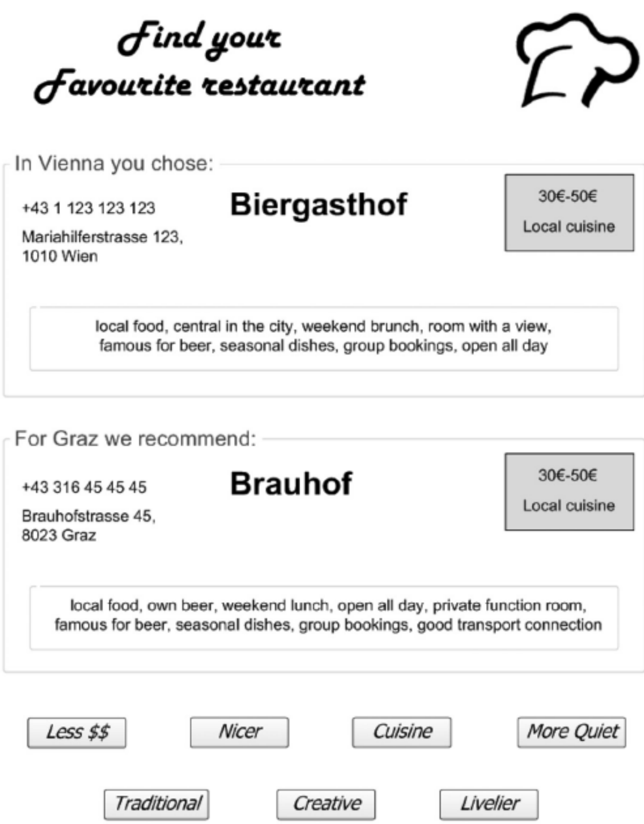


Compound critiques

- Changing one value at a time might be tedious
- Compound critiques allows multiple changes
  - "Increase prize and quality"



More critiquing types



- Critiquing
  - Similarity-based navigation in item space
- Unit critiquing
  - Critiquing of single properties
- Compound critiques
  - Critiquing of multiple properties
- Dynamic critiques
  - Critique options only available if applicable
  - Mining of frequent critique patterns
- Incremental critiques
  - Considers critiquing history
- Experience-based critiquing
  - Exploit past interactions that were successful

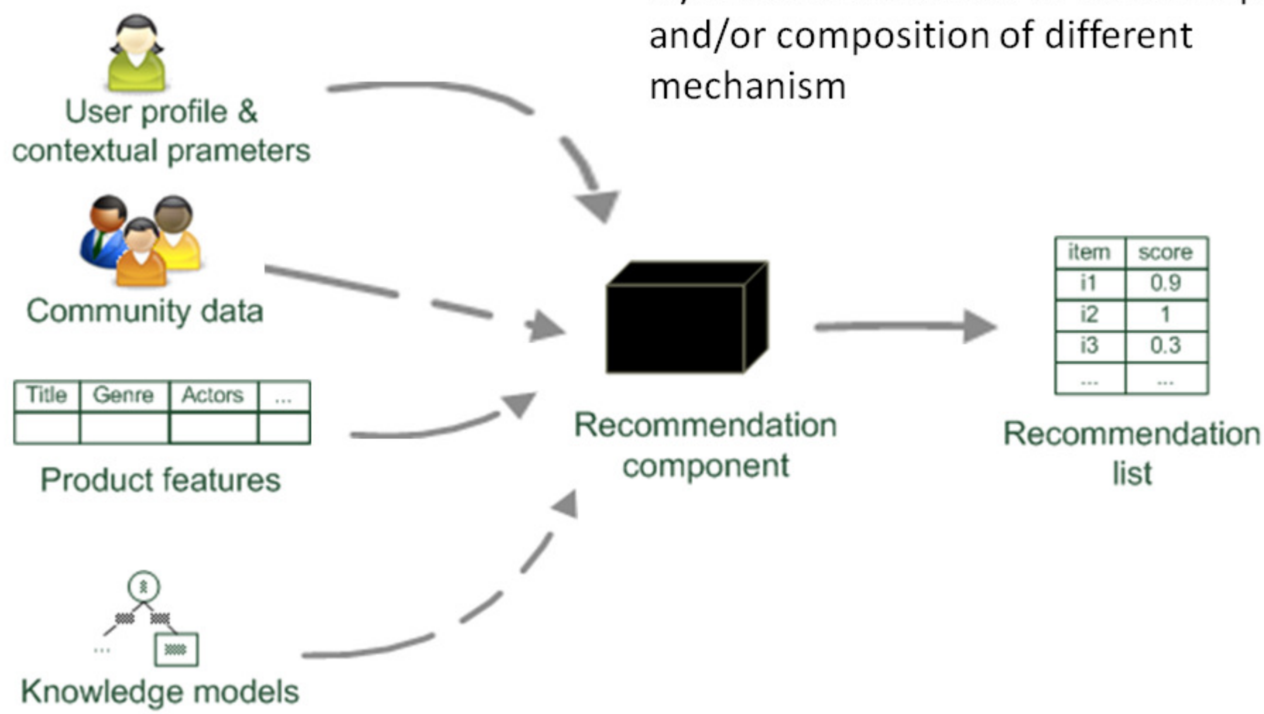
Summary

- Search approaches
  - Query-based → constraint-based recommendation
  - Navigation-based → case-based (critiquing-based) recommendation
- Knowledge-based recommendation
  - Constraint-based: goal is to fulfill a given set of constraints
  - Case-based: similarity-based search
  - Both approaches based on similar user interactions
- User support
  - Different types of defaults
  - Ranking of candidate items on the basis of MAUT
- Consistency management
  - Conflict sets: not fulfillable combinations of constraints (minimality property)
  - Diagnoses: show how to resolve conflicts (minimality property)

Limitations of knowledge-based recommendation

- Cost of knowledge acquisition
  - From domain experts
  - From users
  - From web resources
- Accuracy of preference models
  - Very fine granular preference models require many interaction cycles
  - Collaborative filtering models preference implicitly
- Independence assumption can be challenged
  - Preferences are not always independent from each other
  - But additive models such as MAUT assume independent preferences

Hybrid approaches



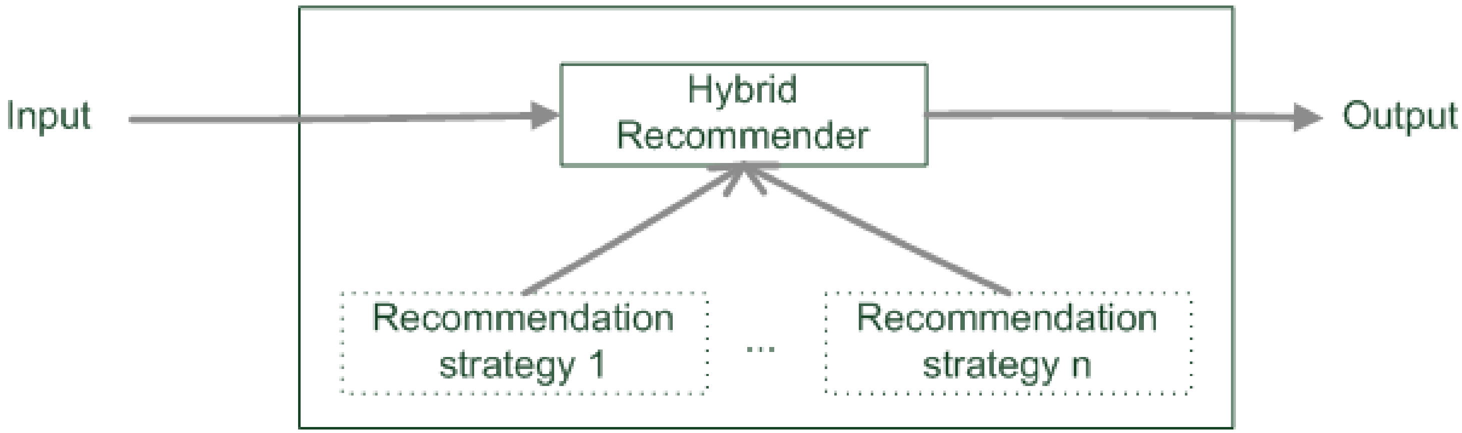
Hybrid: combinations of various inputs and/or composition of different mechanism

Hybrid recommender systems

- Collaborative filtering, content-based filtering, knowledge-based recommendation
  - All pieces of information can be relevant in real-world advisory or recommendation scenarios
  - But all have their shortcomings
- Idea of crossing two (or more) species/implementations
  - hybrida [lat.]: denotes an object made by combining two different elements
  - Avoid some of the shortcomings
  - Reach desirable properties not (or only inconsistently) present in parent individuals
- Different hybridization designs
  - Monolithic exploiting different features
  - Parallel use of several systems
  - Pipelined invocation of different systems

Monolithic hybridization design

- Only a single recommendation component



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

Monolithic hybridization designs: Feature combination

- "Hybrid" user features:
  - Social features: Movies liked by user
  - Content features: Comedies liked by user, dramas liked by user
  - Hybrid features: users who like many movies that are comedies, …

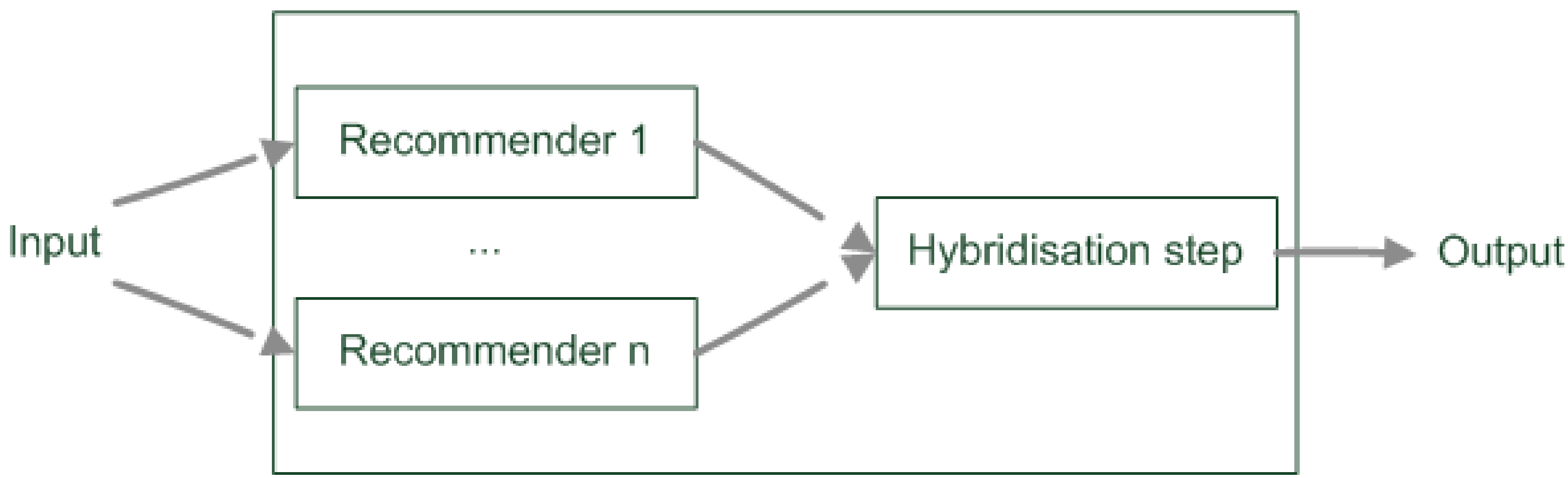
  - "*the common knowledge engineering effort that involves inventing good features to enable successful learning*"

Monolithic hybridization designs: Feature augmentation

- Content-boosted collaborative filtering
  - Based on content features additional ratings are created
  - E.g. Alice likes Items 1 and 3 (unary ratings)
    - Item7 is similar to 1 and 3 by a degree of 0,75
    - Thus Alice likes Item7 by 0,75
  - Item matrices become less sparse
  - Significance weighting and adjustment factors
    - Peers with more co-rated items are more important
    - Higher confidence in content-based prediction, if higher number of own ratings
- Recommendation of research papers
  - Citations interpreted as collaborative recommendations
  - Integrated in content-based recommendation method

Parallelized hybridization design

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



Parallelized design: Weighted

- Compute weighted sum: $rec_{weighted}(u,i) = \sum_{k=1}^{n} \beta_k \times rec_k(u,i)$

Recommender 1		
Item1	0.5	1
Item2	0	
Item3	0.3	2
Item4	0.1	3
Item5	0	

Recommender 2		
Item1	0.8	2
Item2	0.9	1
Item3	0.4	3
Item4	0	
Item5	0	

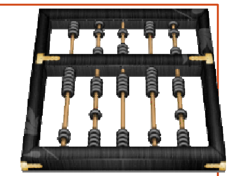
Recommender weighted(0.5:0.5)		
Item1	0.65	1
Item2	0.45	2
Item3	0.35	3
Item4	0.05	4
Item5	0.00	

Parallelized hybridization design: Weighted

- BUT, how to derive weights?
  - Estimate, e.g. by empirical bootstrapping
  - Dynamic adjustment of weights
- Empirical bootstrapping
  - Historic data is needed
  - Compute different weightings
  - Decide which one does best
- Dynamic adjustment of weights
  - Start with for instance uniform weight distribution
  - For each user adapt weights to minimize error of prediction

Parallelized hybridization design: Weighted

- Let's assume Alice actually bought/clicked on items 1 and 4
  - Identify weighting that minimizes Mean Absolute Error (MAE)



Absolute errors and MAE

Beta1	Beta2		rec1	rec2	error	MAE
0.1	0.9	Item1	0.5	0.8	0.23	0.61
		Item4	0.1	0.0	0.99	
0.3	0.7	Item1	0.5	0.8	0.29	0.63
		Item4	0.1	0.0	0.97	
0.5	0.5	Item1	0.5	0.8	0.35	0.65
		Item4	0.1	0.0	0.95	
0.7	0.3	Item1	0.5	0.8	0.41	0.67
		Item4	0.1	0.0	0.93	
0.9	0.1	Item1	0.5	0.8	0.47	0.69
		Item4	0.1	0.0	0.91	

- **MAE improves as *rec2* is weighted more strongly**

$$MAE = \frac{\sum_{r_i \in R} \sum_{k=1}^{n} \beta_k \times |rec_k(u,i) - r_i|}{|R|}$$

Parallelized design: Weighted

- BUT: didn't rec1 actually rank Items 1 and 4 higher?

Recommender 1		
Item1	0.5	1
Item2	0	
Item3	0.3	2
Item4	0.1	3
Item5	0	

Recommender 2		
Item1	0.8	2
Item2	0.9	1
Item3	0.4	3
Item4	0	
Item5	0	

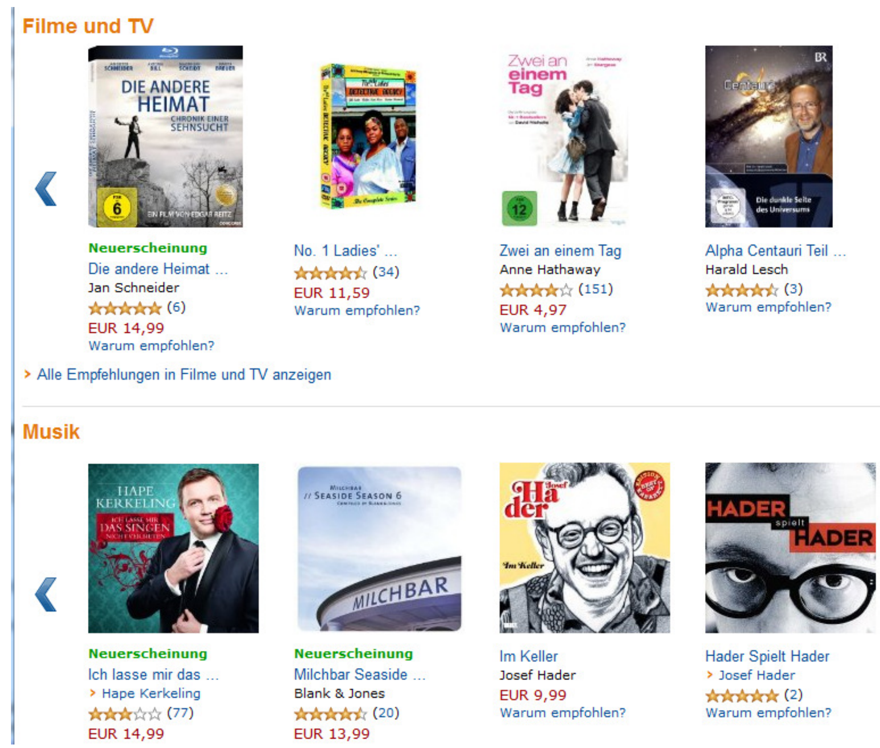
- Be careful when weighting!
  - Recommenders need to assign comparable scores over all users and items
    - Some score transformation could be necessary
  - Stable weights require several user ratings

Parallelized design: Switching

- Special case of dynamic weights
  (all weights except one are 0)
- Requires an oracle that decides which recommender should be used
- Example:
  - Ordering on recommenders and switch based on some quality criteria:
    - If too few ratings in the system, use knowledge-based, else apply collaborative filtering
  - More complex conditions based on contextual parameters possible; classification techniques can be applyied

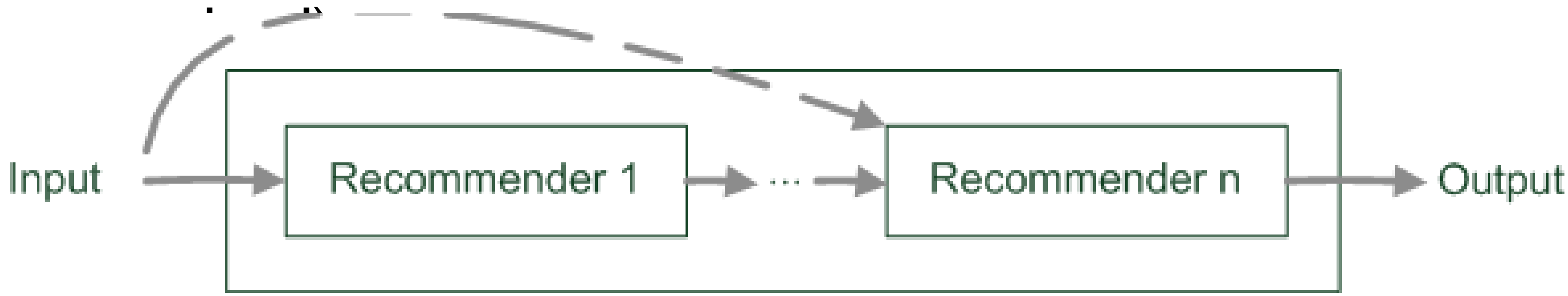
Parallelized design: Mixed

- Combines the results of different recommender systems at the level of user interface
- Results of different techniques are presented together
- Recommendation result for user u and item i is the set of tuples $< score, k >$ for each of its n constituting recommenders rec_k



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



Pipelined hybridization designs: Cascade

Recommender 1		
Item1	0.5	1
Item2	0	
Item3	0.3	2
Item4	0.1	3
Item5	0	

Recommender 2		
Item1	0.8	2
Item2	0.9	1
Item3	0.4	3
Item4	0	
Item5	0	

Recommender cascaded (rec1, rec2)		
Item1	0,80	1
Item2	0,00	
Item3	0,40	2
Item4	0,00	
Item5	0,00	

- Recommendation list is continually reduced
- First recommender excludes items
  - Remove absolute no-go items (e.g. knowledge-based)
- Second recommender assigns score
  - Ordering and refinement (e.g. collaborative)

Pipelined hybridization designs: Meta-level

- Successor exploits a model Δ built by predecessor

$$rec_{meta-level}(u,i) = rec_n(u,i,\Delta_{rec_{n-1}})$$

- $\Delta_{rec_{n-1}}$ is model built by RS_{n-1} exploited by RS_n

- Examples:
  - Fab: content-based, collaborative recommendation
    - Online news domain
    - Contend based recommender builds user models based on weighted term vectors
    - Collaborative filtering identifies similar peers based on weighted term vectors but makes recommendations based on ratings
  - Collaborative, constraint-based meta-level RS
    - Collaborative filtering identifies similar peers
    - A constraint base is learned by exploiting the behavior of similar peers
    - Learned constraints are employed to compute recommendations

Limitations and success of hybridization strategies

- Only few works that compare strategies from the meta-perspective
  - Most datasets do not allow to compare different recommendation paradigms
    - i.e. ratings, requirements, item features, domain knowledge, critiques rarely available in a single dataset
  - Thus, few conclusions that are supported by empirical findings
    - Monolithic: some preprocessing effort traded for more knowledge included
    - Parallel: requires careful matching of scores from different predictors
    - Pipelined: works well for two antithetic approaches
- Netflix competition – "stacking" recommender systems
  - Weighted design based on >100 predictors – recommendation functions
  - Adaptive switching of weights based on user model, parameters (e.g. number of ratings in one session)

Recommender Systems
An introduction

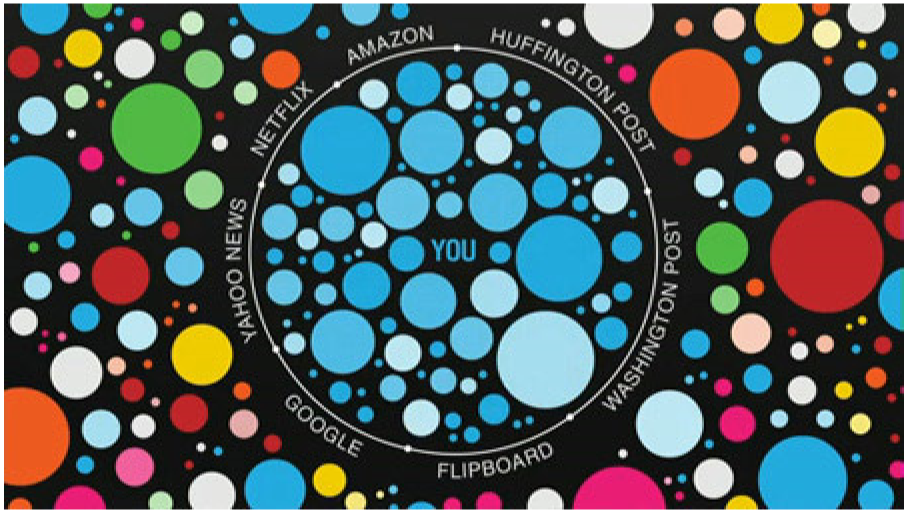
Dietmar Jannach, TU Dortmund, Germany

Slides presented at PhD School 2014, University Szeged, Hungary

dietmar.jannach@tu-dortmund.de

The Filter Bubble

- Are we inside a filter bubble?



- View and discuss…

Explaining recommendations

Explanations in recommender systems

Motivating example

- "The digital camera *Profishot* is a must-buy for you because . . . ."
- Why should recommender systems deal with explanations at all?
- In e-commerce settings, 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

Explanations at Amazon.de

Why recommended ?

Because you bought …

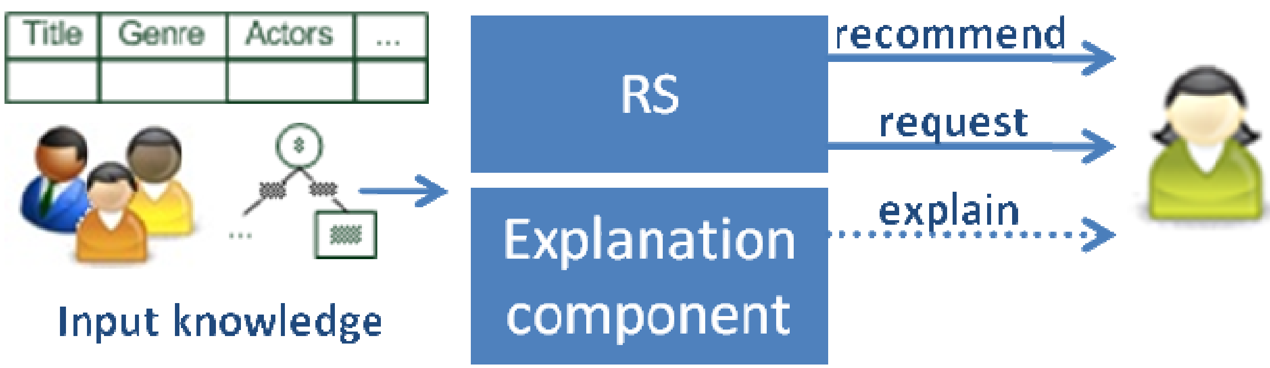
Do not use for recommendations

What is an Explanation?

- “A piece of information exchanged in a communication process”
- Brewer et al. (1998) distinguishes between
  - **functional,**
    - "The car type Jumbo-Family-Van of brand Rising-Sun would be well suited to your family because you have four children and the car has seven seats"
  - **causal,**
    - "The light bulb shines because you turned it on"
  - **intentional,**
    - "I washed the dishes because my brother did it last time"
    - "You have to do your homework because your dad said so"
  - **and scientific explanations**
    - Express relations between the concepts formulated in various scientific fields and are typically based on refutable theories

Explanations in recommender systems

Additional information to explain the system's output following some objectives



Goals when providing explanations (1)

- Transparency
  - Provide information so the user can comprehend the reasoning used to generate a specific recommendation
  - Provide information as to why one item was preferred over another
- Validity
  - Allow a user to check the validity of a recommendation
  - Not necessarily related to transparency
    - E.g., a neural network (NN) decides that product matches to requirements
    - Transparent disclosure of NN's computations will not help, but a comparison of required and offered product features allows customer to judge the recommendation's quality.

Goals when providing explanations (2)

- Trustworthiness
  - Trust building can be viewed as a mechanism for reducing the complexity of human decision making in uncertain situations
  - Reduce the uncertainty about the quality of a recommendation
- Persuasiveness
  - Persuasive explanations for recommendations aim to change the user's buying behavior
  - E.g., a recommender may intentionally dwell on a product's positive aspects and keep quiet about various negative aspects
- Effectiveness
  - The support a user receives for making high-quality decisions
  - Help the customer discover his or her preferences
  - Help users make better decisions

Goals when providing explanations (3)

- Efficiency
  - Reduce the decision-making effort
  - Reduce the time needed for decision making
  - Another measure might also be the perceived cognitive effort
- Satisfaction
  - Improve the overall satisfaction stemming from the use of a recommender system
- Relevance
  - Additional information may be required in conversational recommenders
  - Explanations can be provided to justify why additional information is needed from the user

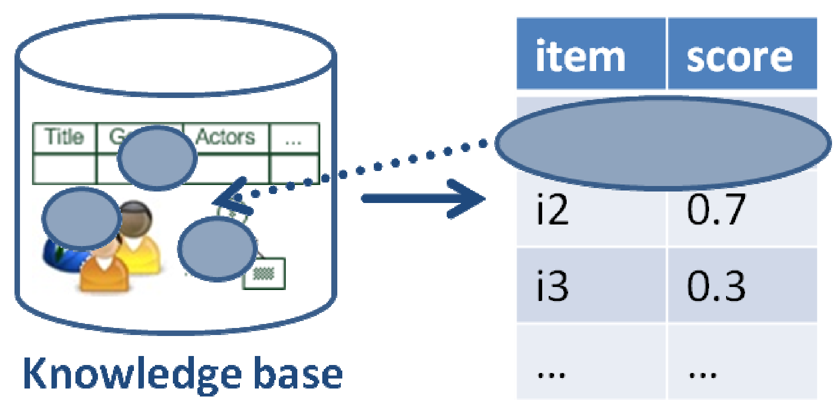
Goals when providing explanations (4)

- Comprehensibility
  - Recommenders can never be sure about the knowledge of their users
  - Support the user by relating the user's known concepts to the concepts employed by the recommender
- Education
  - Educate users to help them better understand the product domain
  - Deep knowledge about the domain helps customers rethink their preferences and evaluate the pros and cons of different solutions
  - Eventually, as customers become more informed, they are able to make wiser purchasing decisions
- The aforementioned aims for generating explanations can be interrelated
  - Persuasiveness**+** $\rightarrow$ Trust**-**
  - Effectiveness**+** $\rightarrow$ Trust**+**
  - …

Explanations in general

- *How?* and *Why?* explanations in expert systems
- Form of abductive reasoning
  - Given: $KB \vDash_{RS} i$ (item i is recommended by method RS)
  - Find $KB' \subseteq KB$ s.t. $KB' \vDash_{RS} i$
- Principle of succinctness
  - Find smallest subset of $KB' \subseteq KB$ s.t. $KB' \vDash_{RS} i$

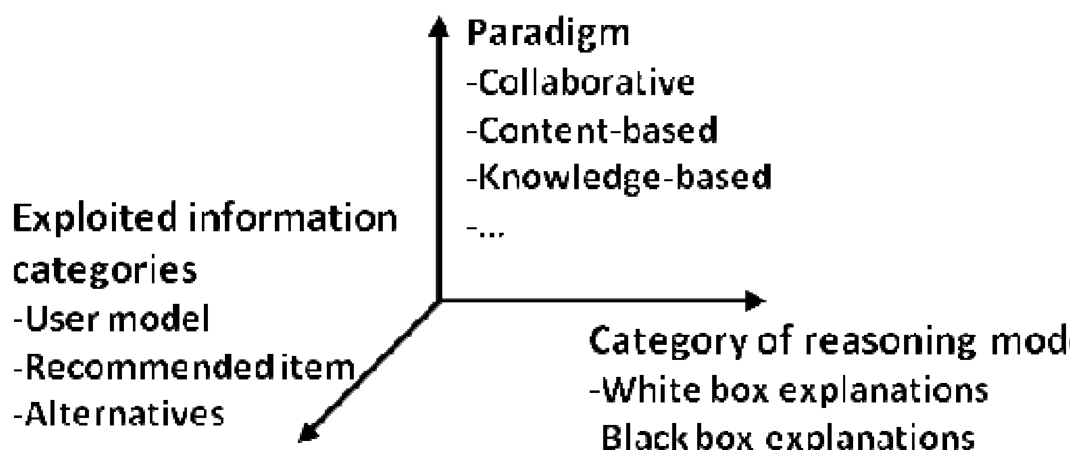
    i.e. for all $KB'' \subset KB'$ holds $KB'' \nvDash_{RS} i$
- But additional filtering
  - Some parts relevant for deduction, might be obvious for humans



Taxonomy for generating explanations

Major design dimensions of current explanation components:

- Category of reasoning model for generating explanations
  - White box
  - Black box
- RS paradigm for generating explanations
  - Determines the exploitable semantic relations
- Information categories



Explanations in CF recommenders

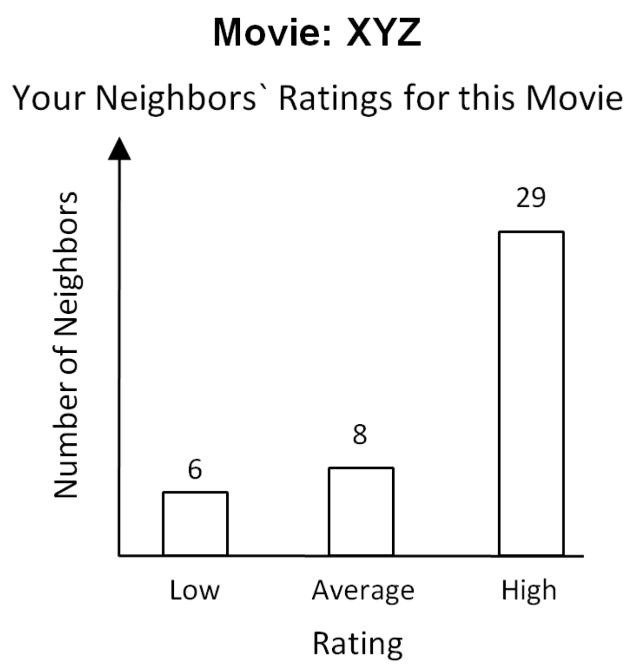
- Explicit recommendation knowledge is not available
- Recommendations based on CF cannot provide arguments as to why
  - a product is appropriate for a customer or
  - why a product does not meet a customer's requirements
- The basic idea of CF is to mimic the human word-of-mouth recommendation process
- Therefore, give a comprehensible account of how this word-of-mouth approach works:
  - Customers rate products
  - The CF locates customers with similar ratings (i.e., tastes), called neighbors
  - Products that are not rated by a customer are rated by combining the ratings of the customer’s neighbors

Evaluating explanation interfaces
(Herlocker et al. 2000)

- Herlocker et al. (2000) examined various implementations of explanation interfaces for the MovieLens Systems
  - Twenty-one variants were evaluated
- User study design / questionnaire
  - 21 different explanation approaches
  - Users were asked to rate on a 1-7 scale
    - how likely they would be to go to see a recommended movie given the explanation
  - Base case with no explanation included
  - Additional interface using past performance
    - "MovieLens has provided accurate predictions for you 80% of the time in the past"

Study results

- The best-performing explanation interfaces are based on the ratings of neighbors



Movie: XYZ

Personalized Prediction: ****

Your Neighbors` Ratings for this Movie

Rating	Number of Neighbors
★	2
★★	4
★★★	8
★★★★	20
★★★★★	9

- Similar neighbors liked the recommended film, and this was comprehensibly presented.
  - The histogram performed better than the table

Study results

- Recommenders using the simple statement about the past performance of MovieLens
  - The second best performer!
- Content-related arguments mentioning the similarity to other highly rated films or a favorite actor or actress
  - Among the best performers
- Poorly designed explanation interfaces decreased the willingness of customers to follow the recommendation
  - Even compared with the base case
- Too much information has negative effects
  - Poor performance was achieved by enriching the data presented in histograms with information about the proximity of neighbors
- Supporting recommendations with ratings from domain authorities, such as movie critics:
  - No increase in acceptance

Explanations for CB / KB recommenders

- Content-based
  - Properties characterizing items
  - TF*IDF model

- Knowledge based
  - Properties of items
  - Properties of user model
  - Additional mediating domain concepts

Content-based techniques

- Could be based on item similarity
  - Because you liked …
  - Similar items …
    - Amazon.com's list labels convey explanatory information
- Hybrid techniques
  - Combine ratings with content information
    - Keyword-style explanations
    - Tag-based explanations
    - Tag clouds

Keyword-style explanations

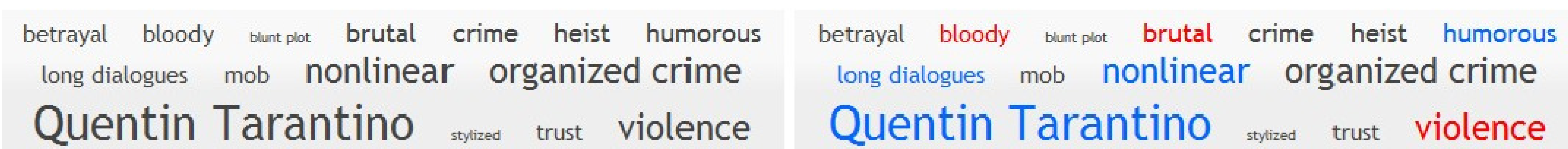
- Can be more effective than rating-based ones

Word	Strength	Explain
thriller	36.19	Explain
paris	30.13	Explain
spy	21.28	Explain
action	18.92	Explain
identity	18.72	Expl
conspiracy	16.53	Expl
killer	13.26	Expl

The word action is positive due to the movie ratings:

Movie	Rating	Occurrence
Sin City	5	29
Casino Royale	4	3

"Tagsplanations" and tag clouds



Explanations in case-based RS

- The generation of solutions in case-based recommenders is realized by identifying the products that best fit a customer's query
  - Based on item features and a similarity measure
- Each item of a product database corresponds to a case
- Customer query puts constraints on the attributes of products
  - For example, a customer is interested only in digital cameras that cost less than a certain amount of money

Explanations in case-based RS

- In particular, given a query Q about a subset A_Q of attributes A of a case (product) description, the similarity of a case C to Q can be defined defined as

$$sim(C,Q) = \sum_{a \in A_Q} w_a sim_a(C,Q)$$

- The function $sim_a(C,Q)$
  - describes the similarity of the attribute values of the query Q and the case C for the attribute a
- This similarity is weighted by w_a, expressing the importance of the attribute to the customer
- A recommendation set is composed of all cases C that have a maximal similarity to the query Q

Explaining solutions (1)

- A possible approach to answer a "why-question" is to compare the presented case with the customer requirements
  - highlight which constraints are fulfilled and which are not
- Example:

id	price	mpix	Opt-zoom	LCD-size	movies	sound	waterproof
p1	148	8.0	4x	2.5	no	no	yes
p2	182	8.0	5x	2.7	yes	yes	no
p3	189	8.0	10x	2.5	yes	yes	no
p4	196	10.0	12x	2.7	yes	no	yes
p5	151	7.1	3x	3.0	yes	yes	no
p6	199	9.0	3x	3.0	yes	yes	no
p7	259	10.0	3x	3.0	yes	yes	no
p8	278	9.1	10x	3.0	yes	yes	yes

Explaining solutions (2)

- If a customer is interested in digital cameras with a price less than 150, then p1 is recommended.

id	price	mpix	Opt-zoom	LCD-size	movies	sound	waterproof
p1	148	8.0	4x	2.5	no	no	yes
p2	[illegible]	8.0	5x	2.7	yes	yes	no
p3	[illegible]	8.0	10x	2.5	yes	yes	no
p4	196	10.0	12x	2.7	yes	no	yes
p5	151	7.1	3x	3.0	yes	yes	no
p6	199	9.0	3x	3.0	yes	yes	no
p7	259	10.0	3x	3.0	yes	yes	no
p8	278	9.1	10x	3.0	yes	yes	yes

Why?

Explaining solutions (3)

- The weights of the attributes can be incorporated into the answers
  - If the customer requires a price less than 160 and LCD size of more than 2.4 inches, where LCD size is weighted much more than price, then p5 is recommended

id	price	mpix	Opt-zoom	LCD-size	movies	sound	waterproof
p1	148	8.0	4x	2.5	no	no	yes
p2	182	8.0	5x	2.7	yes	yes	no
p3	189	8.0	10x	2.5	yes	yes	no
p4	196	10.0	12x	2.7	yes	no	yes
p5	151	7.1	3x	3.0	yes	yes	no
p6	199	9.0	3x	3.0	yes	yes	no
p7	2[illegible]	[illegible]	[illegible]	[illegible]	yes	yes	no
p8	278	9.1	10x	3.0	yes	yes	yes

Why?

Explaining solutions (4)

- The requirements of a customer might be too specific
  - Why-explanations provide information about the violated constraints
- If the customer requires a price less than 150 and a movie function, then no product fulfills these requirements.

id	price	mpix	Opt-zoom	LCD-size	movies	sound	waterproof
p1	148	8.0	4x	2.5	no	no	yes
p2	182	8.0	5x	2.7	yes	yes	no
p3	189	8.0	10x	2.5	yes	yes	no
p4	196	10.0	12x	2.7	yes	no	yes
p5	151	7.1	3x	3.0	yes	yes	no
p6	199	9.0	3x	3.0	yes	yes	no
p7	259	10.0	3x	3.0	yes	yes	no
p8	278	9.1	10x	3.0	yes	yes	yes

Most similar products

Explaining solutions (5)

- p1 and p5 can be considered as most similar products for a given similarity function
  - although one of the user requirements is not satisfied
- A why-explanation for p1 would be,
  - "p1 is within your price range but does not include your movie requirement."
- Automated techniques can be used to
  - generate minimal sets of customer requirements that explain why no products fit, or to
  - to propose minimal changes to the set of requirements such that matching products exist

Explanations in constraint-based RS

- CSP-based and other reasoning-based systems
  - The constraints that determine a certain value for a specific variable can be identified
  - Inference tracing
  - Natural language explanations can be automatically constructed
    - e.g., based on annotated constraints
    - However,
      - There might be multiple reasons for a variable assignment
      - Not all of them are relevant

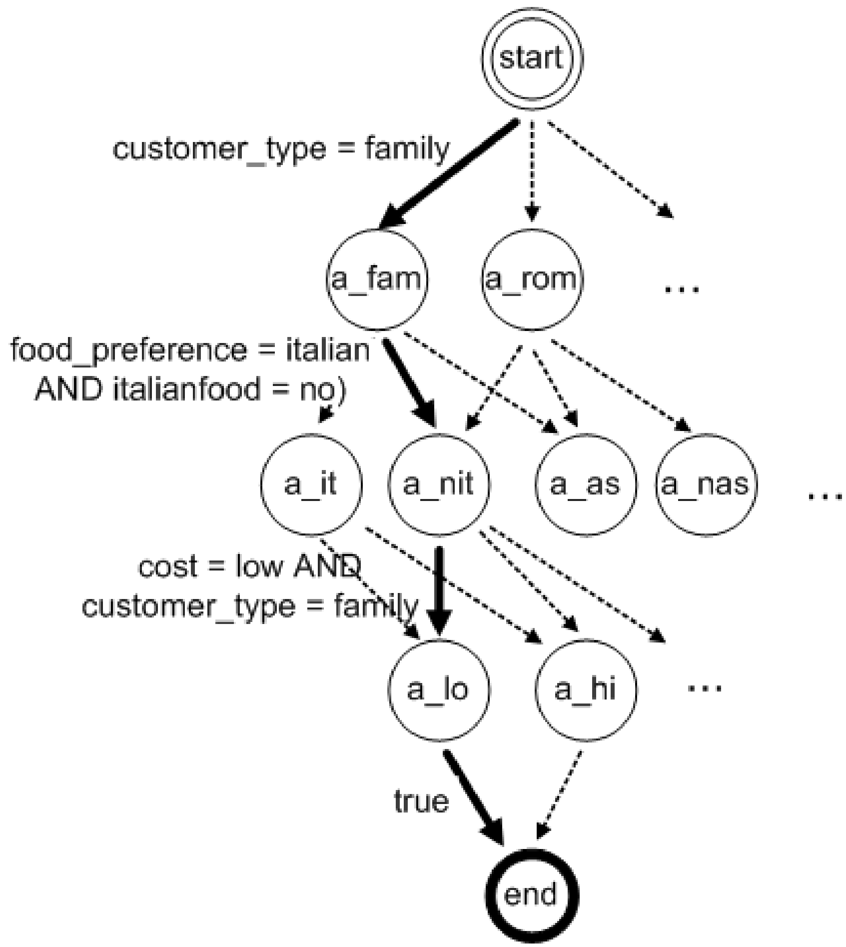
An argumentation-based approach

- Explanation as a sequence of arguments

$$e = \langle a_1, \ldots, a_n \rangle$$

  - e is natural language text and every $a \in e$ can be a textual phrase
- Model: 5-tuple ($X_I \cup X_U$, D, C, Q, E)
  - X ... Finite set of variables
    - X_I item variables
    - X_U user variables
  - C ... Constraints
  - Q ... States/arguments
  - E ... Transitions
- Transitions $a_1.c.a_2$ connect two arguments $a_1, a_2 \in Q$ with a constraint c.
- The functions start(Q) and end(Q) return the start and the end state.

Representation of the model



- Knowledge-based explanation model
  - represented by a layered directed acyclic graph (DAG)
  - Contains a distinguished start and an end node
  - Each layer presentes a property of the item

Example

X_I = {food_preference,...}

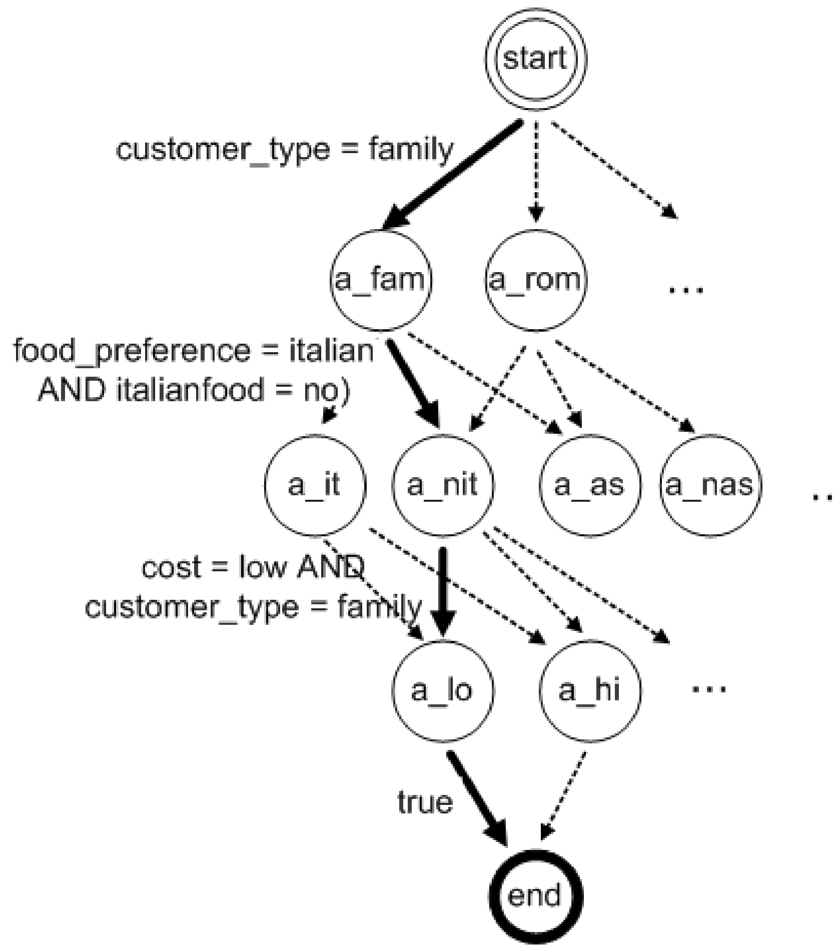
X_U = {food_served,...}

D =dom(customer_type)
={family, couple},

C = {c1: customer_type= family, ...}

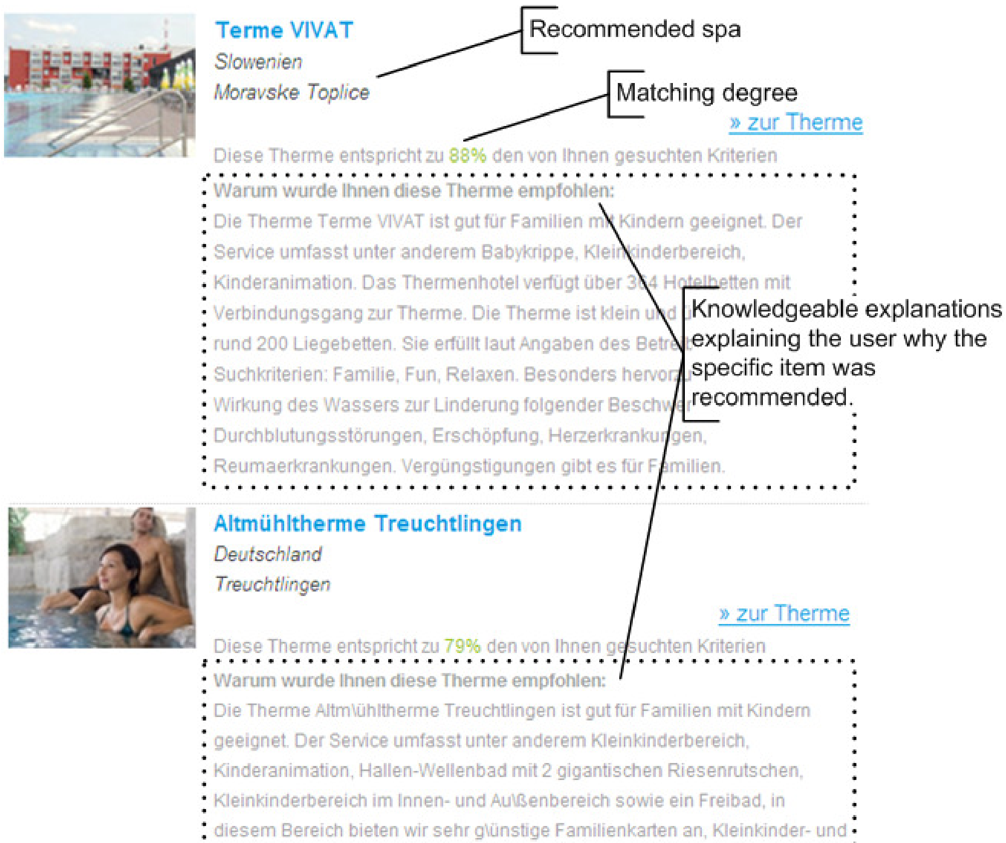
Q = {start, a_fam, ..., end}

E = {start.c1.a_fam, ...}

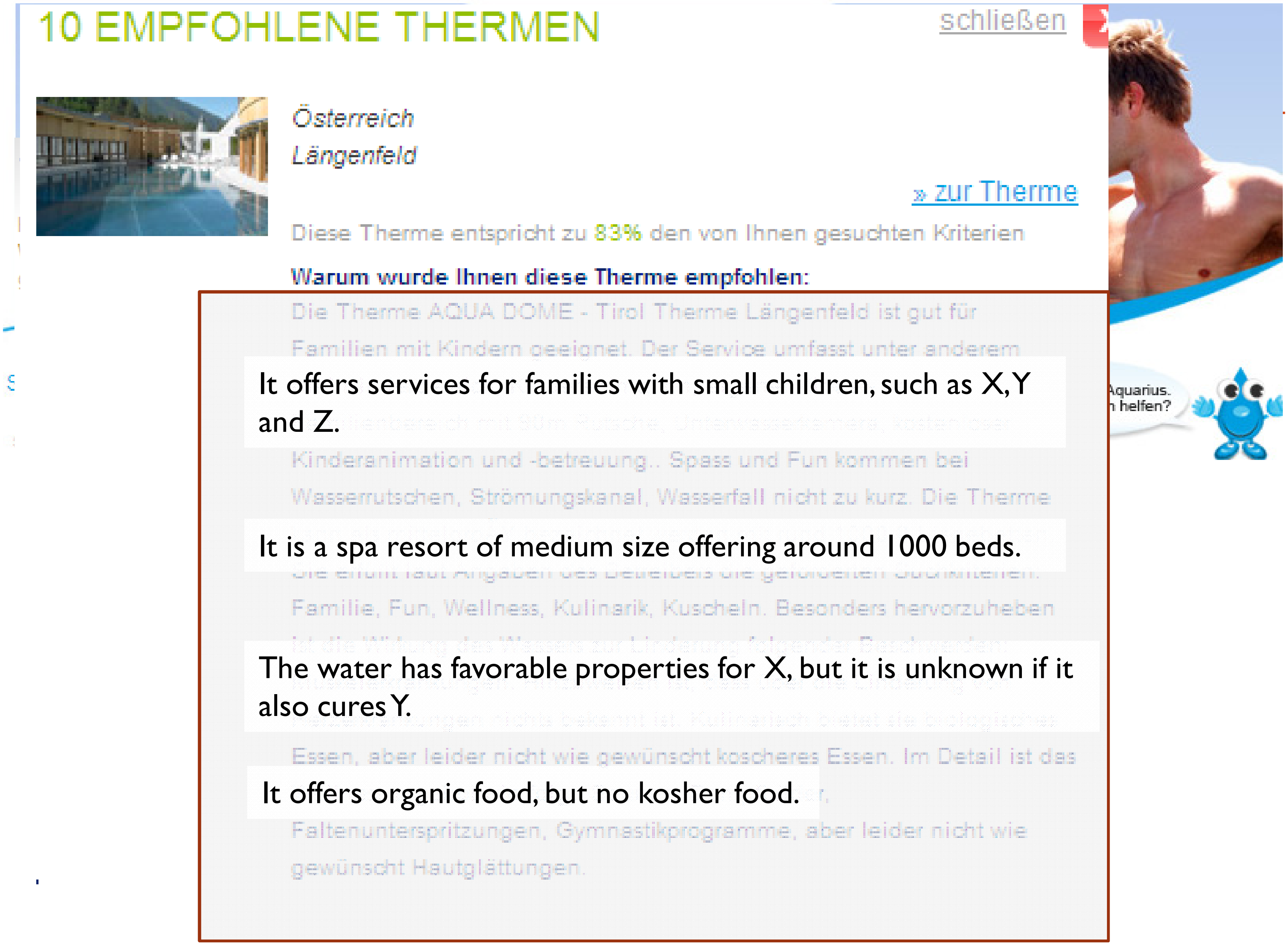


- Bold faced transitions provide a valid sequence of arguments <start, a_{fam}, a_{it}, a_{lo}, end>

Presentation of the explanation



- Users receive different explanations for each recommended item (here: spa resort)



It offers services for families with small children, such as X, Y and Z.

It is a spa resort of medium size offering around 1000 beds.

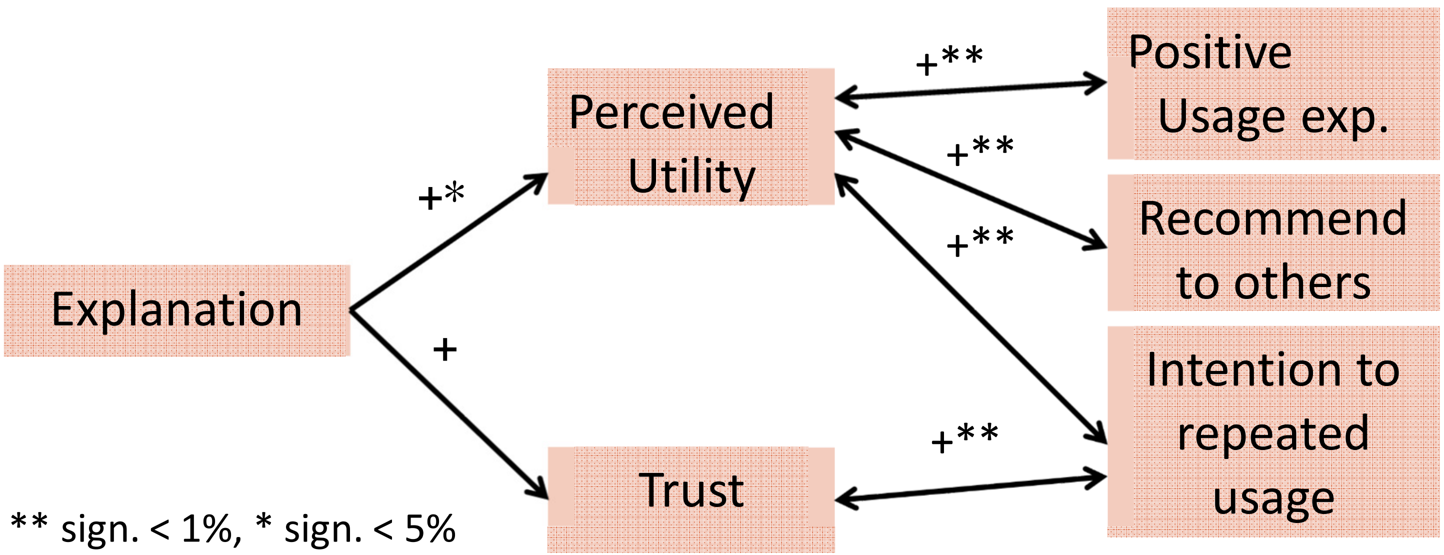
The water has favorable properties for X, but it is unknown if it also cures Y.

It offers organic food, but no kosher food.

Evaluation

- Methodology
  - Online test on real-world platform
    - (see http://www.thermencheck.com)
  - More then 200 participants
  - Randomly division of the participants into two groups:
    - Group A: explanations for the recommendation were shown
    - Group B: no explanation was shown
    - Questionnaire after interaction
  - Questions
    - usability and the use of the system
    - the intention to repeated use,
    - positive usage experience and
    - willingness to recommend to others

Results for the explanation feature

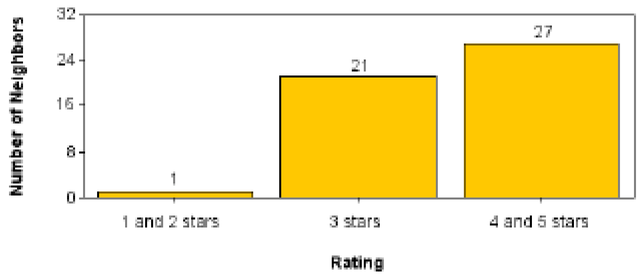


- Knowledgeable explanations significantly **increase the** users' perceived utility
- Perceived utility strongly correlates with usage intention etc.

An example for a laboratory study

How should I explain?

- Recent study on different explanation interfaces
  - Gedikli et al., IJHCS 2014
- Compared 10 different styles
  - Including rating-based ones, (personalized) tag clouds, and simple aggregating techniques
  - Involved over 100 participants



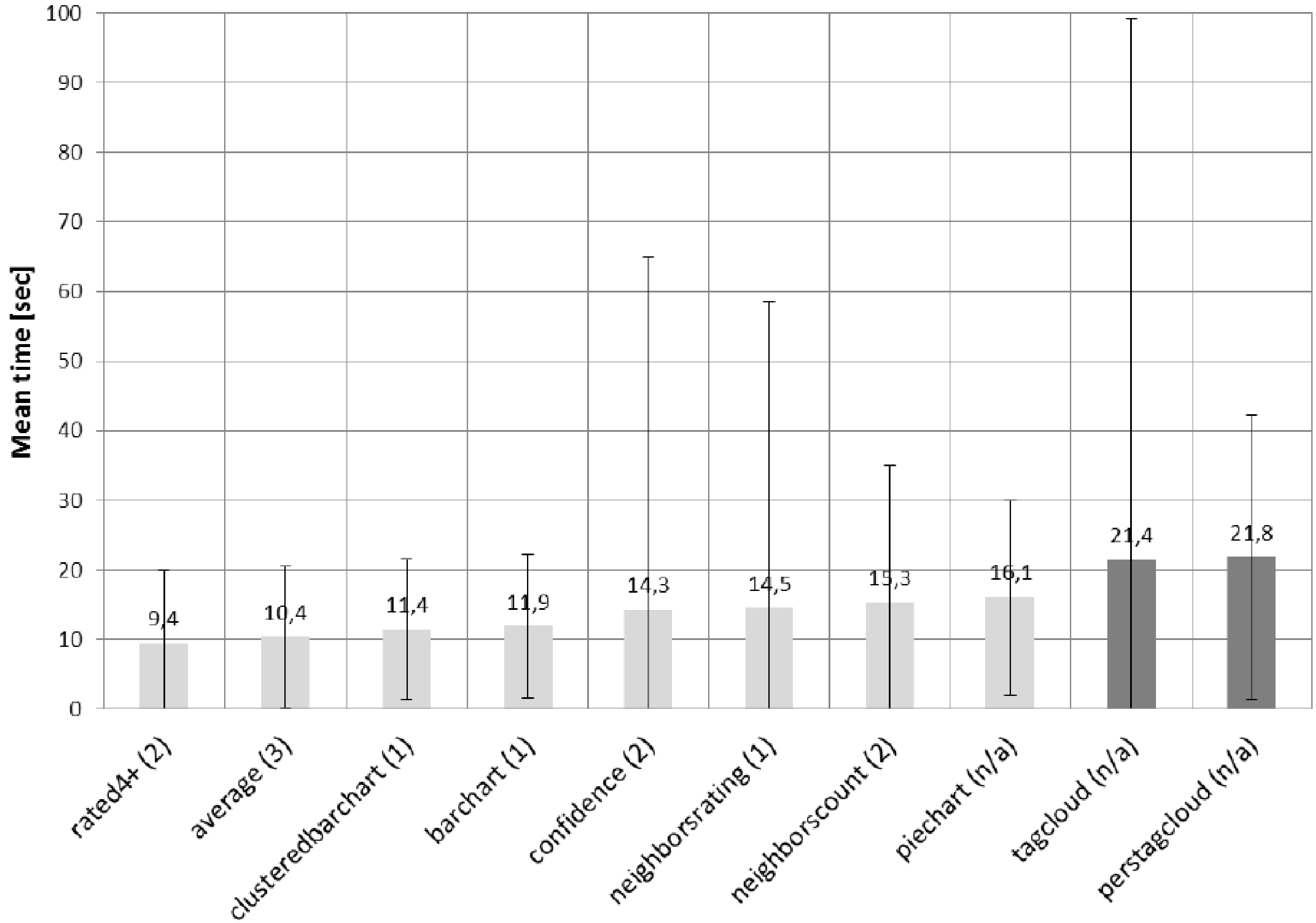
Experimental goals and procedure

- Measurement dimensions
  - Efficiency:
    - Can users decide faster?
  - Effectiveness:
    - Is their rating based on explanations similar to the rating without explanations
  - Persuasiveness:
    - Do the explanations induce a bias?
  - Trade offs?

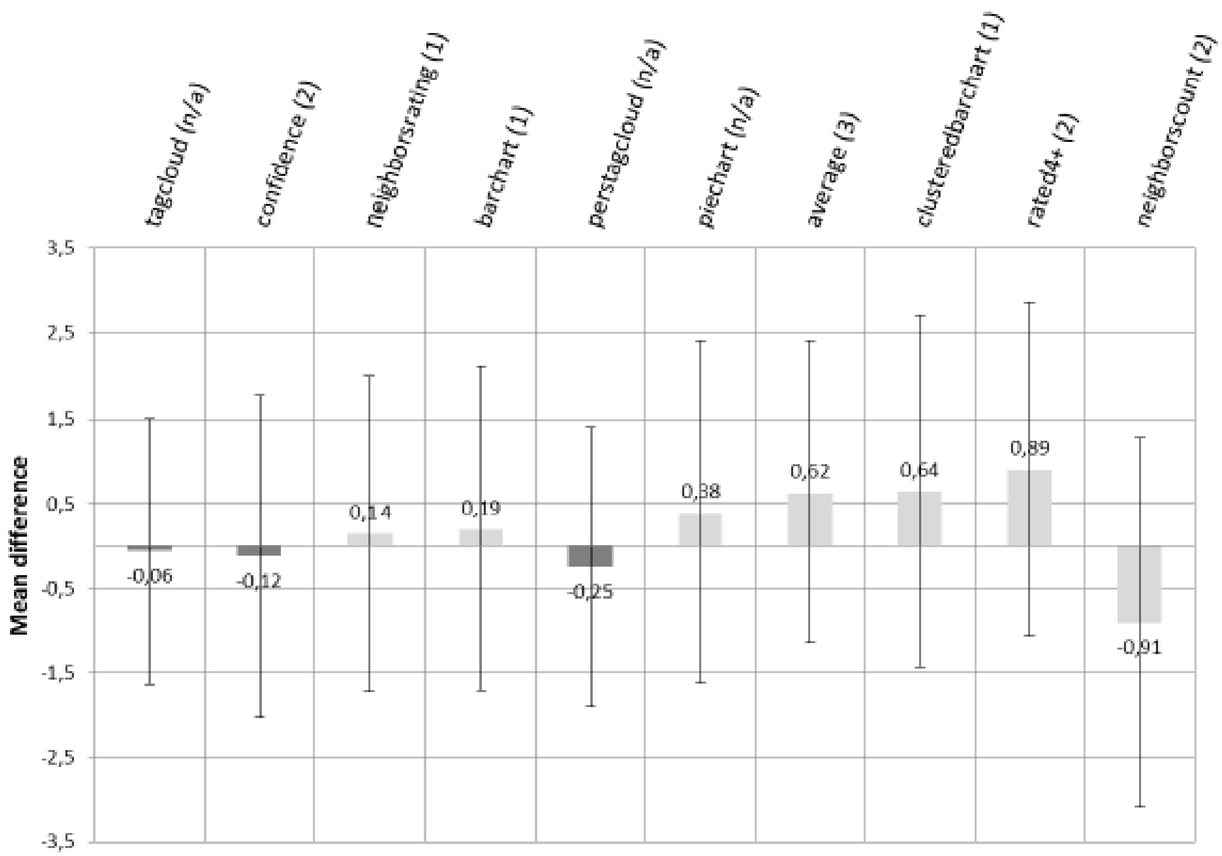
Procedure 1 Experimental procedure used in the laboratory study

```
Get sample ratings from the user.
R = Set of recommendations for the user.
E = Set of explanation interfaces.
for all randomly chosen (r, e) in R x E do
    Present explanation using interface e for recommendation r to the user.
    Ask the user to rate r and measure the time taken by the user.
end for
for all recommendation r in R do
    Show detailed information about r to the user.
    Ask the user to rate r again.
end for
Ask the user to rate the explanation interfaces.
```

Results – mean time for deciding



Effectiveness and persuasion



Results – Perceived transparency

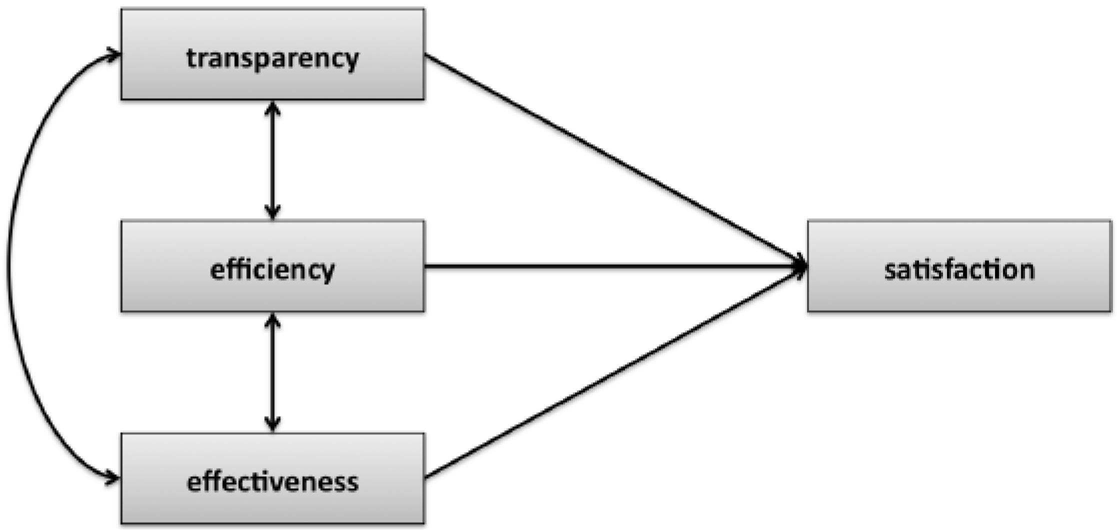
#		transparency ▽	Std Dev	Herlocker Group
1	perstagcloud	5.61	1.53	-
2	barchart	5.51	1.26	1
3	piechart	5.41	1.21	-
4	clusteredbarchart	5.40	1.25	1
5	rated4+	5.27	1.28	2
6	neighborsrating	5.12	1.13	1
7	average	5.07	1.46	3
8	tagcloud	**5.05**	1.60	-
9	confidence	**4.65**	1.34	2
10	neighborscount	**2.80**	1.70	2

Results - Satisfaction

#		satisfaction ▽	Std Dev	Herlocker Group
1	perstagcloud	4.96	1.93	-
2	average	4.70	1.39	3
3	rated4+	4.63	1.50	2
4	tagcloud	4.59	1.91	-
5	clusteredbarchart	4.57	1.60	1
6	barchart	4.56	1.40	1
7	confidence	4.45	1.39	2
8	piechart	4.32	1.75	-
9	neighborsrating	**3.95**	1.46	1
10	neighborscount	**2.09**	1.38	2

Results – Relationships between variables

- Path analysis



- Observations
  - Transparency has a positive effect on satisfaction
  - Efficiency and effectiveness have no strong effect on satisfaction

Explanations in RS: Summary

- Various types of explanations exist
- Different goals possible
- Possible types of explanations
  - depend on available information and recommendation approach
- Explanations may be used to shape the wishes and desires of customers but are a double-edged sword
  - Explanations can help the customer to make wise buying decisions,
  - But, explanations can also be abused to push a customer in a direction which is advantageous solely for the seller

Recommender Systems
An introduction

Dietmar Jannach, TU Dortmund, Germany

Slides presented at PhD School 2014, University Szeged, Hungary

dietmar.jannach@tu-dortmund.de

Selected topics in RS

- What is hot (now and in the last years), emerging?
  - A subjective and unsorted selection

Context-awareness

- Increased interest in the last years in the community
- What I want to watch depends …
  - Alone or with family or friends?
  - In the afternoon or late at night?
  - On weekdays or the weekend?
  - How is my current mood and interest?
    - Documentary, intellectual movie or blockbuster?
  - Looking for the freshest one available?
  - Want to see a movie that is similar to one I saw last week?

Context-awareness: Challenges in research

- Recently proposed approaches
  - Mostly extend existing technique that, e.g.,
    - Factor in additional context variables like time into the models, or
    - Filter or re-rank recommendations based on contextual parameters
  - Often use small datasets
    - Time or geographic location (taken from Social Web sources) as known factors
  - Techniques and findings sometimes comparably simple like "recommend nearby events"
  - Sometimes limited reproducibility
    - Specific, non-public datasets

Social and Semantic Web Recommender Systems

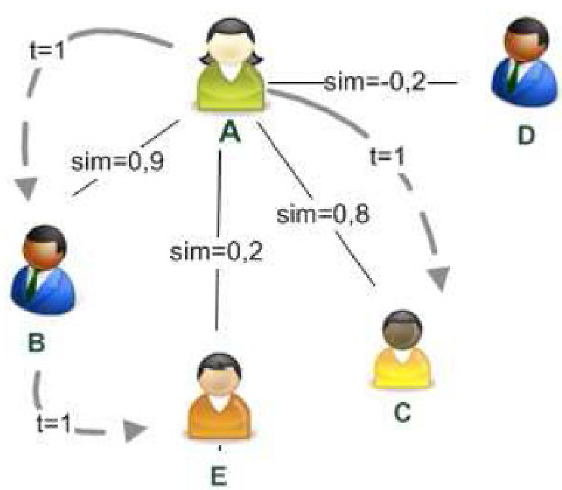
- Social Web perspectives
  - Make recommendations **for** "resources" on the Social Web
    - Friends, photos, web sites, tweets, posts, news, groups, <u>tags</u>, …
    - News filtering and ranking
      - Filter bubble?
  - Make recommendations **based on** information from Social Web
    - Use the social graph to find like-minded usere
    - Use information from posts, tweets etc to estimate user preferences
    - Develop trust-based recommendations
- Semantic Web
  - Build better "content-based" systems
    - Linked Data, Semantic Web databases, Wikipedia/DBPedia

Trust-aware recommender systems

- Explicit trust statements between users
  - can be expressed on some social web platforms (epinions.com)
  - could be derived from relationships on social platforms
  - Trust is a multi-faceted, complex concept
  - Goes however beyond an "implicit" trust notion based on rating similarity
    - Some papers simply see similarity as indicator for trust …
- Exploiting trust information in RS
  - to improve accuracy (neighborhood selection)
  - to increase coverage
  - could be used to make RS robust against attacks

Early Trust-based System

- Input
  - rating matrix
  - explicit trust network (ratings between 0 – no trust, and 1 – full trust)
- Prediction
  - based on usual weighted combination of ratings of the nearest neighbors
  - similarity of neighbors is however based on the trust value



Note:
- Assume standard Pearson CF with min. 3 peers and similarity-threshold = 0.5
- No recommendation for A possible
- However: Assuming that trust is transitive, also the rating of E could be used
- Good for cold-start situations

Social trust algorithms

- Trust-propagation
  - Various algorithms and propagation schemes possible (including global "reputation" metrics
- Recommendation accuracy
  - Hybrids combining similarity and trust shown to be more accurate in some experiments
- Symmetry and Distrust
  - Trust is not symmetric
  - How to deal with explicit distrust statements?
    - If A distrusts B and B distrusts – what does this tell us about A's relation to C?
- Evaluation
  - Accuracy improvements possible ; increase of coverage
  - Not many publicly available data sets

Tags and Folksonomies

- Collaborative tagging in the Web 2.0
  - Users add tags to resources (such as images)
  - Folksonomies are based on freely-used keywords (e.g., on flickr.com)
  - Note: not as formal as ontologies, but more easy to acquire
- Folksonomies and Recommender Systems?
  - Use tags to recommend items
  - Use RS technology to recommend tags

Tag-based recommendation

- Tags as content annotations
  - use content-based algorithms to recommend interesting tags
- Possible approach:
  - determine keywords/tags that user usually uses for his highly-rated movies
  - find un-rated movies having similar tags
- Metrics:
  - take keyword frequencies into account
  - compare tag clouds (simple overlap of movie tags and user cloud; weighted comparison)
- Possible improvements:
  - tags of a user can be different from community tags (plus: synonym problem)
  - add semantically related words to existing ones based on WordNet information

Tag-enhanced collaborative filtering

- Difference to content-boosted CF
  - tags/keywords are not "global" annotations, but local for a user
- Possible approach: a combined, tag-aware CF method
  - remember, in user-based CF:
    - similarity of users is used to make recommendations
    - here: view tags as additional items (0/1 rating, if user used a tag or not); thus similarity is also influenced by tags
  - likewise: in item-based CF, view tags as additional users (1, if item was labeled with a tag)
- Predictions
  - combine user-based and item-based predictions in a weighted approach
  - experiments show that only combination of both helps to improve accuracy

Recommending tags

- Remember: Users annotate items very differently
- RS technology can be used to help users find appropriate tags
  - thus, making the annotations of items more consistent
  - Possible approach:
    - Derive two-dimensional projections of User X Tag X Resource data
    - Use nearest-neighbor approach to predict item rating
      - use one of the projections
  - Evaluation
    - User-Tag similarity better than User-Resource
    - differences on different datasets; always better than "most-popular (by resource)"-strategy
- FolkRank:
    - View folksonomy as graph and apply PageRank idea
    - Method outperforms other approaches

Selected topics in RS

- Evaluation aspects
    - User-centric evaluation
    - Multi-metric evaluation
    - Cross-domain recommendation
    - New metrics in offline designs
    - Consideration of biases
- Preferences
    - Preference elicitation,
    - Active learning
- Decision making
    - Consumer psychology, human decision processes
- Case studies
    - More needed, as always, different domains ..

Algoritmic and evaluation topics

- Algorithms
  - Learning to rank
    - Optimize (a proxy of) a rank measure
  - Deep learning
    - e.g., deep neural networks
    - Learning multiple levels of representation / abstraction
  - Scalability
    - Process billions of ratings
    - Distributed architectures
- Data
  - Social Web, mult-criteria ratings
  - Reviews ..

Multi-criteria recommender systems

- Multi-criteria ratings
  - Users can rate items in various dimensions
  - Typical in the hotel domain
    - e.g., TripAdvisor, Booking.com, HRS.com
  - Also on Yahoo!Movies
    - Directing, Acting, Story, ..

- Idea / Problem
  - Can we make more accurate predictions when we know the detailed ratings?
  - Existing approaches
    - I) Use multi-dimensional similarity functions in kNN method
    - II) Learn a importance weights (regression function) to predict the overall rating

Our approach

User	Item	Overall	Service	Value	Rooms
u1	i1	4	2	4	1
u1	i2	2	3	2	4
u1	i3	4	2	4	2
u2	i1	4	4	2	3
u2	i2	2	2	4	3
u2	i3	?	?	?	?

- Learn regression functions *per user* and *per item*
- Use Support Vector Regression to be able to handle the sparse data situation
- Apply feature selection to identify the most important features and to remove noise
- Combine the predictions of the models in a weighted approache
  - Learn optimal weights in the training phase

$$RO_{i1} = w1_{i1} * Service + w2_{i1} * Value + w3_{i1} * Rooms + c_{i1}$$
$$RO_{i2} = w1_{i2} * Service + w2_{i2} * Value + w3_{i2} * Rooms + c_{i2}$$
$$\ldots$$
$$RO_{in} = w1_{in} * Service + w2_{in} * Value + w3_{in} * Rooms + c_{in}$$

Results

- Evaluated on three datasets
  - Hotels
    - HRS.com
    - TripAdvisor.com
  - Movies
    - Yahoo!Movies

Algorithm	HRS-5-5	HRS-3-3	HRS-RAW
SlopeOne	0.68 (1.0)	0.71 (0.99)	0.77 (0.72)
Funk-SVD	0.60 (1.0)	0.64 (0.99)	0.66 (0.73)
MC-Similarity	0.65 (0.32)	0.71 (0.12)	0.77 (0.31)
SV-Regress-I	0.59 (1.0)	0.62 (0.99)	0.72 (0.73)
SV-Regress-U	0.61 (1.0)	0.66 (0.99)	0.66 (0.72)
WeightedSVM	**0.52** (1.0)	**0.56** (0.99)	**0.61** (0.73)

- Measurable accuracy improvements
  - Compared to existing multi-criteria approaches
  - Compared to recent matrix factorization techniques

Other recent topics

- Human descision making
  - Take into account insights from consumer psychology
  - Phenomena like choice overload, (ir)rationality of human descision making processes, preference construction and stability
  - Personality-based recommender systems
- Sales psychology
  - Context effects
    - How to present items
    - Primacy/Recency effects (list positions matter)
    - Decoy effects
  - Trust
- Behavioural patterns
    - Maximizer / Satisfizer

Examples of recent research works

- Popularity and concentration biases of algorithms
- Short-term user interests (Zalando)
- Explanation interfaces for recommenders
- Multi-criteria recommender systems

- Music playlist generation (music recommendation)
  - Discussion of limitations of current evaluation measures
  - Analysis of what makes a good playlist
- Novel applications of recommender systems
  - Process modeling, software development

Thank you!