technische universität dortmund

ALPEN-ADRIA UNIVERSITÄT KLAGENFURT

Introduction to Recommender Systems

Tutorial at ACM Symposium on Applied Computing 2010
Sierre, Switzerland, 22 March 2010

Markus Zanker
University Klagenfurt

Dietmar Jannach
TU Dortmund

About the speakers

- **Markus Zanker**
  - Assistant professor at University Klagenfurt
  - CEO of ConfigWorks GmbH
- **Dietmar Jannach**
  - Professor at TU Dortmund, Germany
- **Research background and interests**
  - Application of Intelligent Systems technology in business
    - Recommender systems implementation & evaluation
    - Product configuration systems
    - Web mining
    - Operations research

Agenda

- **What are recommender systems for?**
  - Introduction
- **How do they work?**
  - Collaborative Filtering
  - Content-based Filtering
  - Knowledge-Based Recommendations
  - Hybridization Strategies
- **How to measure their success?**
  - Evaluation techniques
  - Case study on the mobile Internet
- **Selected recent topics**
  - Attacks on CF Recommender Systems
  - Recommender Systems in the Social Web
- **What to expect?**

Agenda

- **What are recommender systems for?**
  - Introduction
- **How do they work?**
  - Collaborative Filtering
  - Content-based Filtering
  - Knowledge-Based Recommendations
  - Hybridization Strategies
- **How to measure their success?**
  - Evaluation techniques
  - Case study on the mobile Internet
- **Selected recent topics**
  - Attacks on CF Recommender Systems
  - Recommender Systems in the Social Web
- **What to expect?**

Introduction

Problem domain

- **Recommendation systems (RS) help to match users with items**
  - Ease information overload
  - Sales assistance (guidance, advisory, persuasion,…)

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

- **Goal to identify good system implementations**
  - But: multiple optimality criterions exist

Purpose and success criteria (1)

- **Different perspectives/aspects**
  - Depends on domain and purpose
  - No wholistic evaluation scenario exists

- **Retrieval perspective**
  - Reduce search costs
  - Provide ‚correct' proposals
  - Users know in advance what they want

- **Recommendation perspective**
  - Serendipity – identify items from the Long Tail
  - Users did not know about existence

Purpose and success criteria (2)

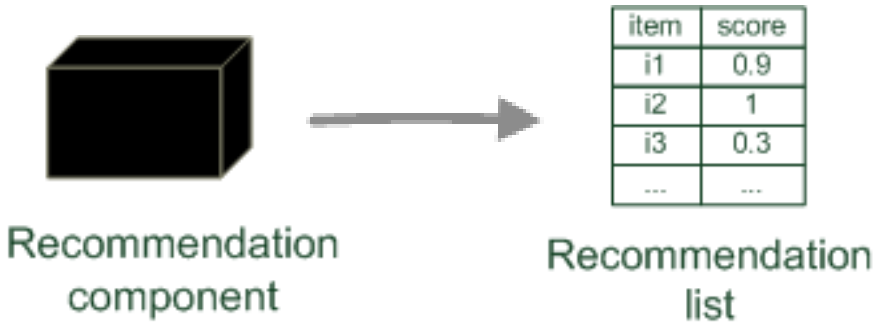
- **Prediction perspective**
  - Predict to what degree users like an item
  - Most popular evaluation scenario in research
- **Interaction perspective**
  - Give users a ‚good feeling‘
  - Educate users about the product domain
  - Convince/persuade users - explain
- **Finally, conversion perspective**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

Recommender systems

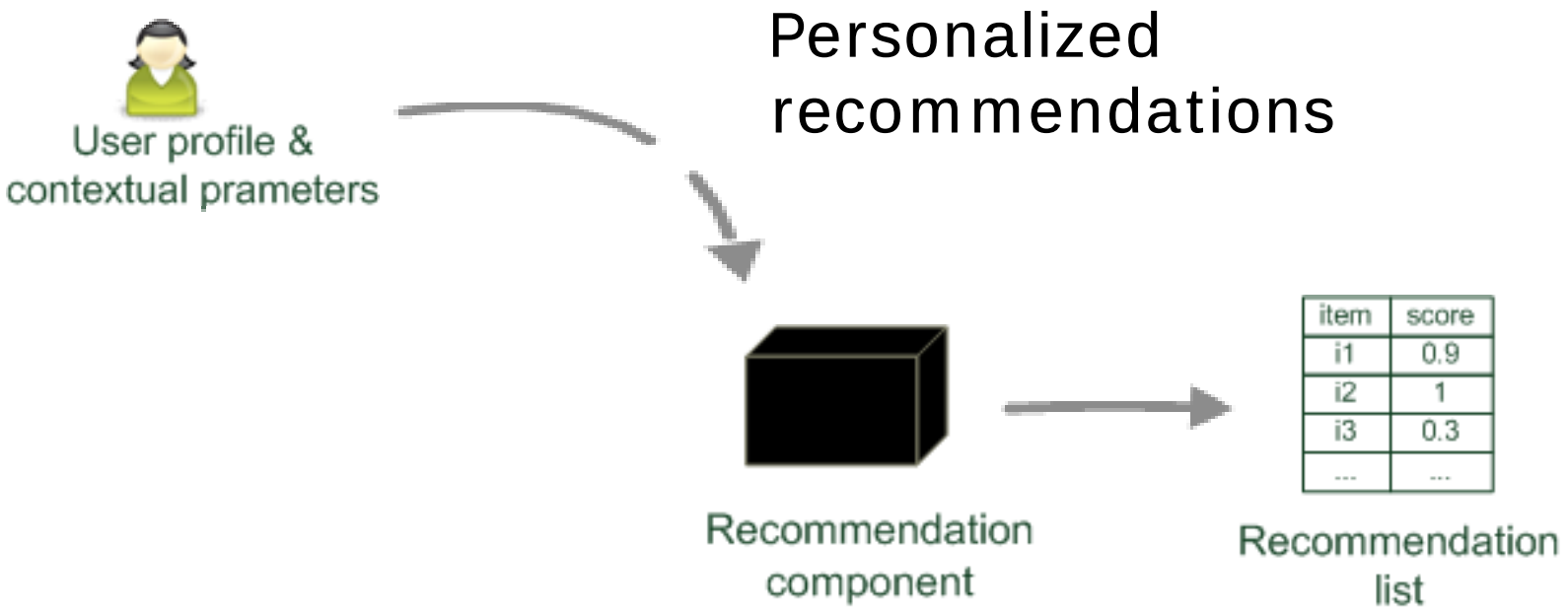
- **RS seen as a function** [AT05]
- **Given:**
  - User model (e.g. ratings, preferences, demographics, situational context)
  - Item
- **Find:**
  - Relevance score
- **Scores responsible for ranking**
- **In practical systems usually not all items will be scored, but task is to find most relevant ones (selection task)**

Paradigms of recommender systems

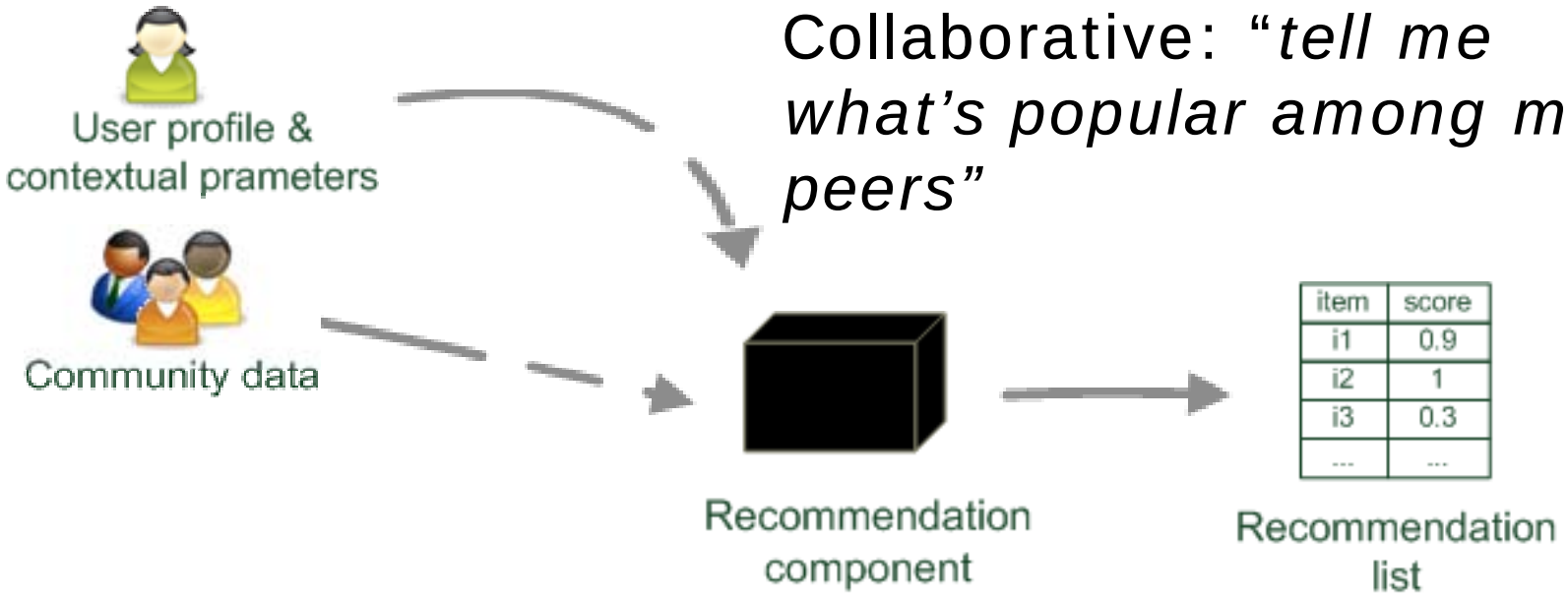
Recommender systems reduce information overload by estimating relevance



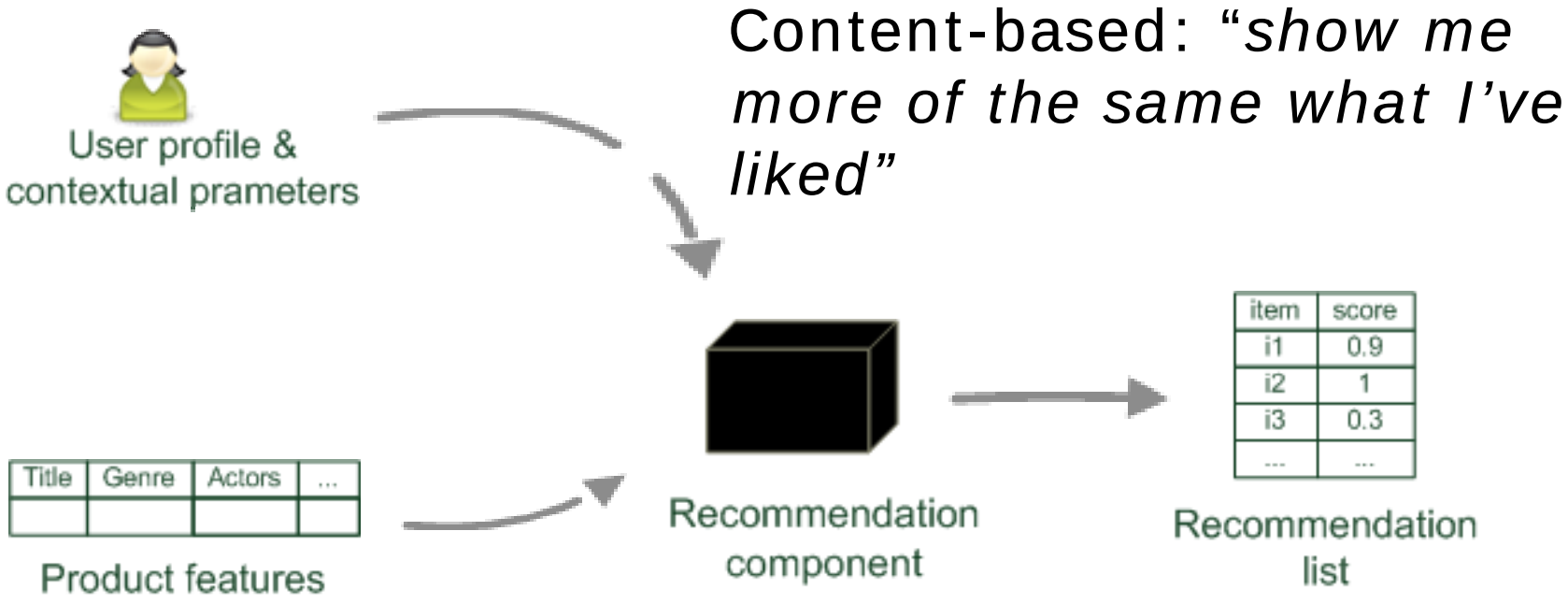
Paradigms of recommender systems



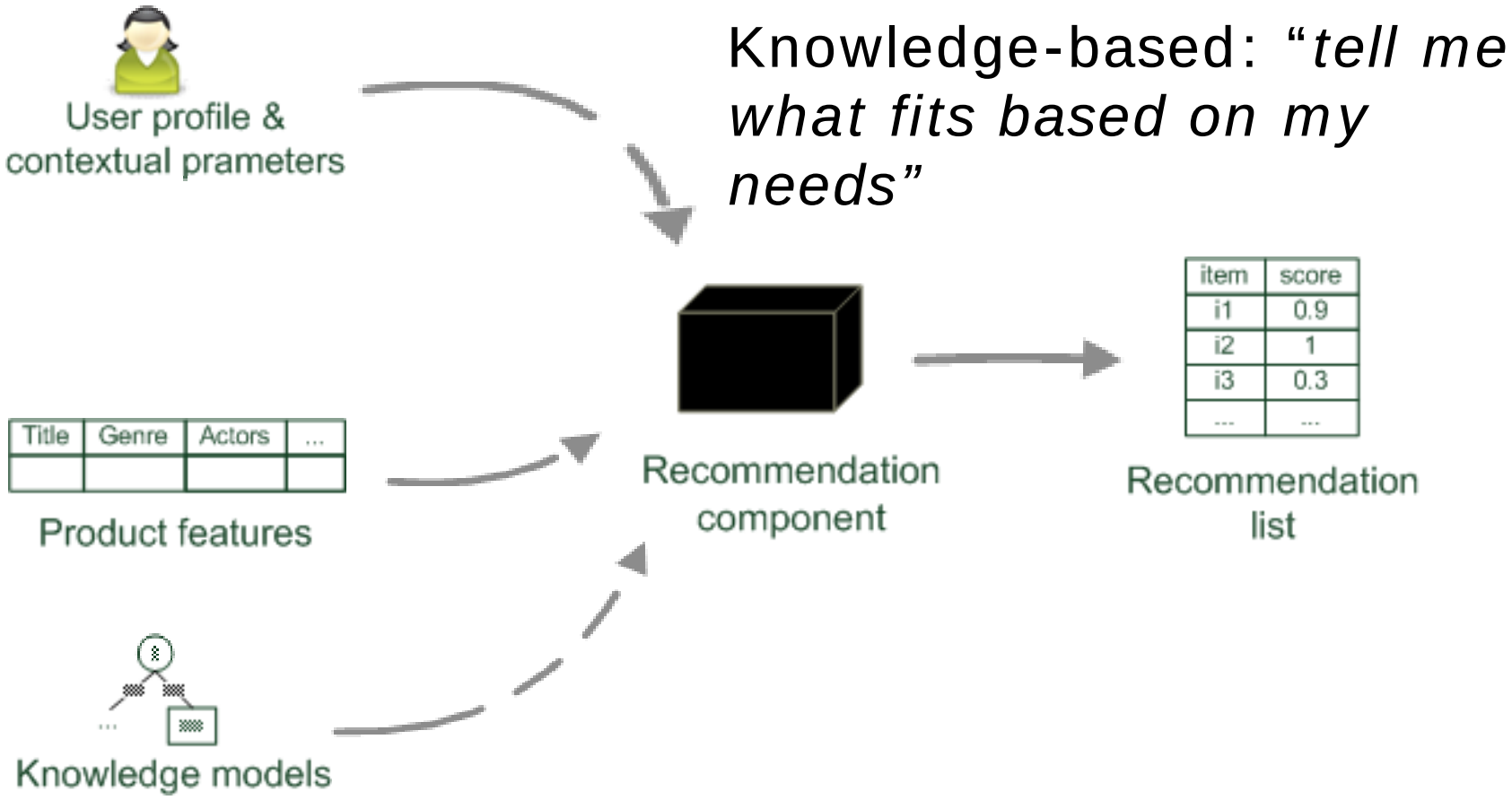
Paradigms of recommender systems



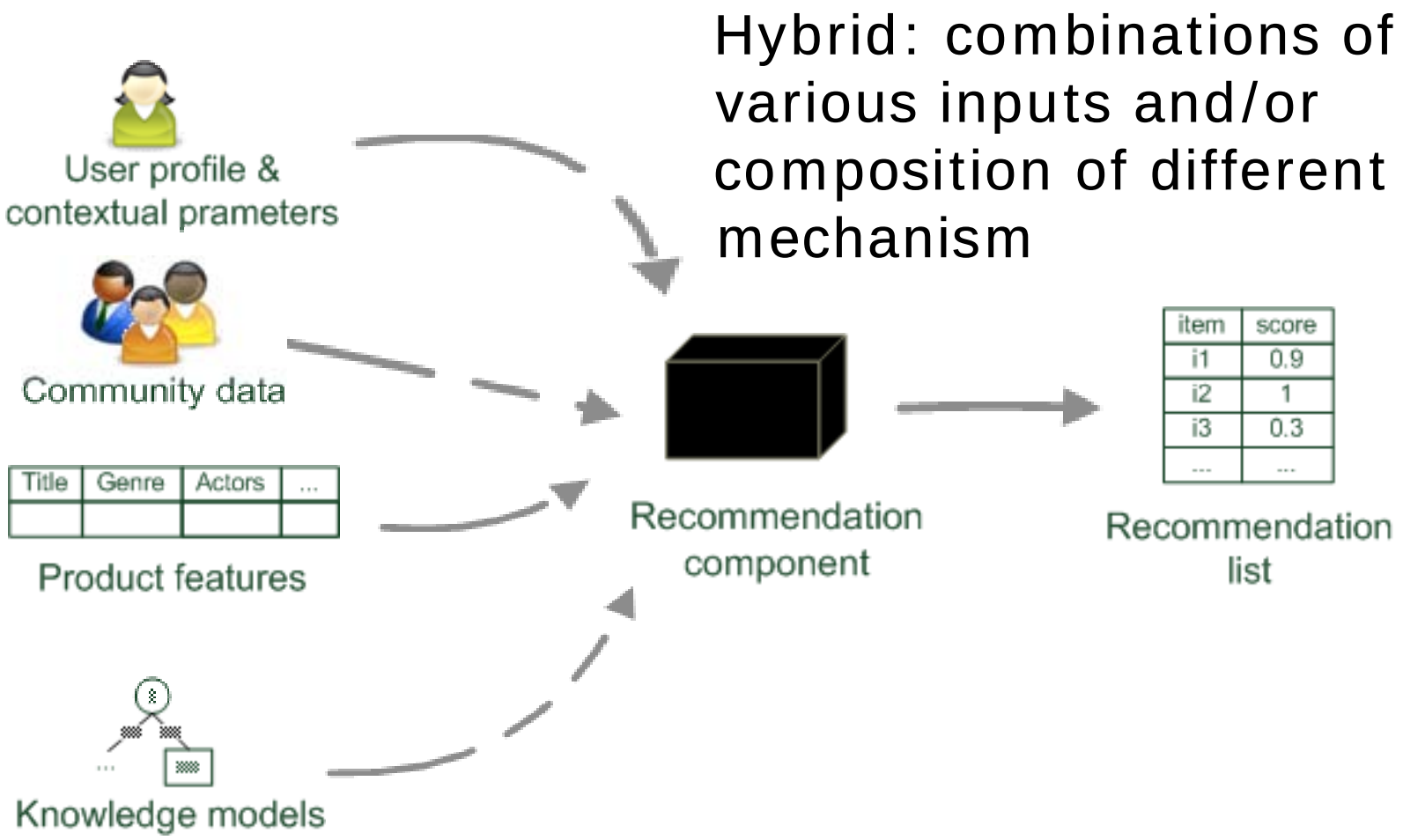
Paradigms of recommender systems



Paradigms of recommender systems



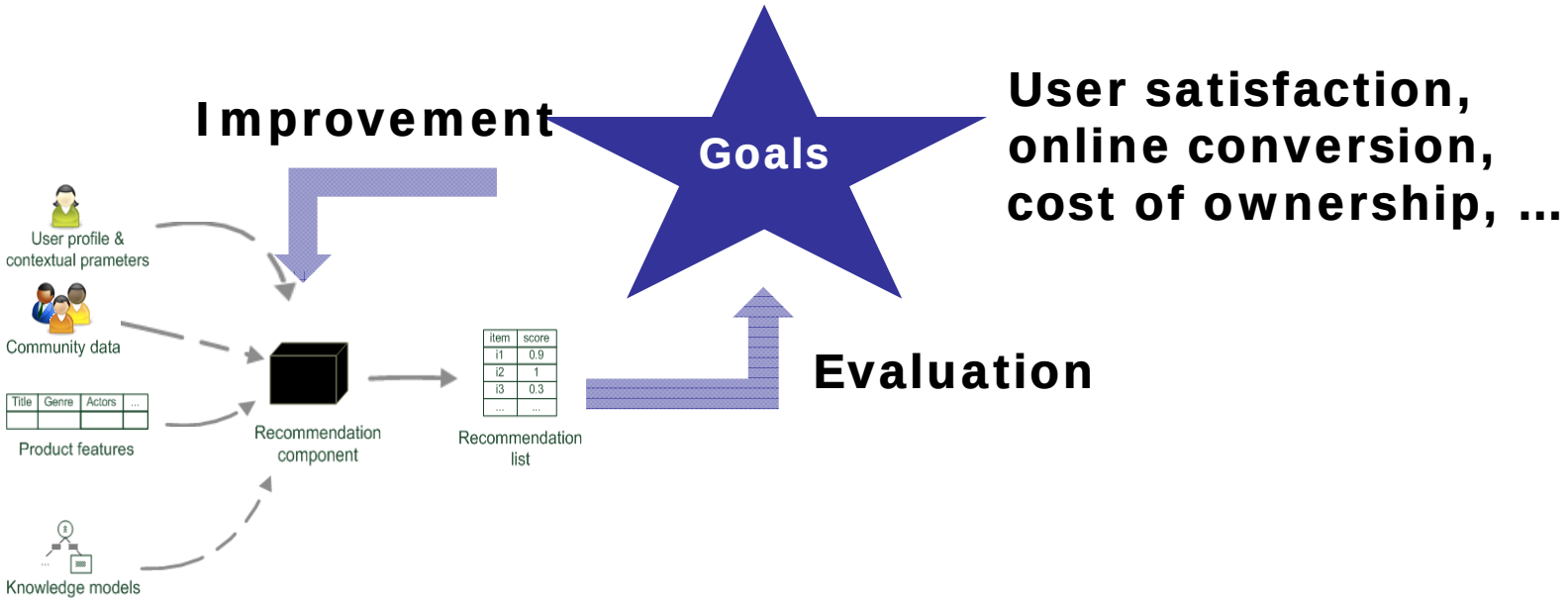
Paradigms of recommender systems



Recommender systems: basic techniques

	Pros	Cons
Collaborative	Nearly no ramp-up effort, serendipity of results, learns market segments	Requires some form of rating feedback, cold start for new users and new items
Content-based	Rating feedback easy to acquire, supports comparisons	Content-descriptions necessary, cold start for new users, no surprises
Knowledge-based	Deterministic rec's, assured quality, no cold-start, can resemble sales dialogue	Knowledge engineering effort to bootstrap, basically static, does not react to short-term trends

Goals of research



- **Explore combinations of collaborative and knowledge-based methods**
  - Hybridization designs
  - Identify domain and user characteristics
- **Feedback loop by empirical evaluations**

Collaborative Filtering

Collaborative Filtering (CF)

- **The most prominent approach to generate recommendations**
  - used by large, commercial e-commerce sites
  - well-understood, various algorithms and variations exist
  - applicable in many domains (book, movies, DVDs, ..)
- **Approach**
  - use the "wisdom of the crowd" to recommend items
- **Basic assumption and idea**
  - Users give ratings to catalog items (implicitly or explicitly)
  - Customers who had similar tastes in the past, will have similar tastes in the future

User-based nearest-neighbor collaborative filtering

- **The basic technique:**
  - Given an "active user" (Alice) and an item I not yet seen by Alice
    - find a set of users (peers) who liked the same items as Alice in the past **and** who have rated item I
    - use, e.g. the average of their ratings to predict, if Alice will like item I
    - do this for all items Alice has not seen and recommend the best-rated

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

ngs?

Measuring user similarity

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

  a, b : users

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

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

  – Possible similarity values between -1 and 1

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

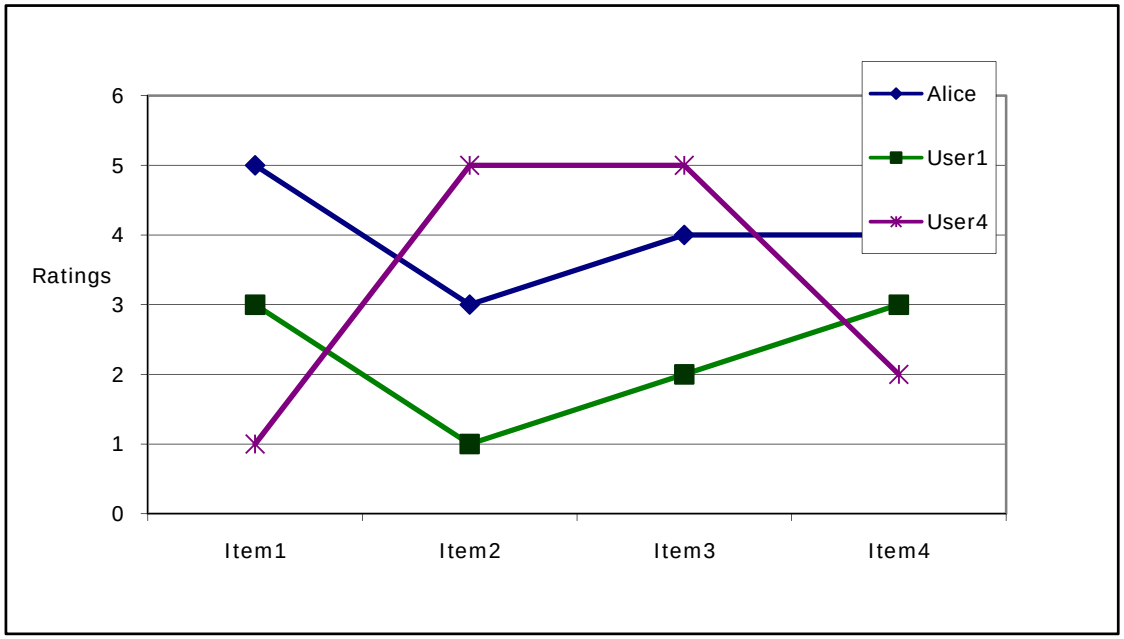
sim = 0,85

sim = 0,70

sim = -0,79

Pearson correlation

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



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

Making predictions

- A common prediction function:

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

- 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 a as a weight
- Add/subtract the neighbors' bias from the active user's average and use this as a prediction

Improving the metrics / prediction function

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

Memory-based and model-based approaches

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

Item-based collaborative filtering

- **Basic idea:**
  - Use the similarity between items (and not users) to make predictions
- **Example:**
  - Look for items that are similar to Item5
  - Take Alice's ratings for these items to predict the rating for Item5

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

The cosine similarity measure

- **Produces better results in item-to-item filtering**
- **Ratings are seen as vector in n-dimensional space**
- **Similarity is calculated based on the angle between the vectors**

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

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

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

Pre-processing for item-based filtering

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

More on ratings

- **Pure CF-based systems only rely on the rating matrix**
- **Explicit ratings**
  - Most commonly used (1 to 5, 1 to 7 Likert response scales)
  - Research topics
    - "Optimal" granularity of scale; indication that 10-point scale is better accepted in movie domain
    - Multidimensional ratings (multiple ratings per movie)
  - Challenge
    - Users not always willing to rate many items; sparse rating matrices
    - How to stimulate users to rate more items?
- **Implicit ratings**
  - clicks, page views, time spent on some page, demo downloads …
  - Can be used in addition to explicit ones; question of correctness of interpretation

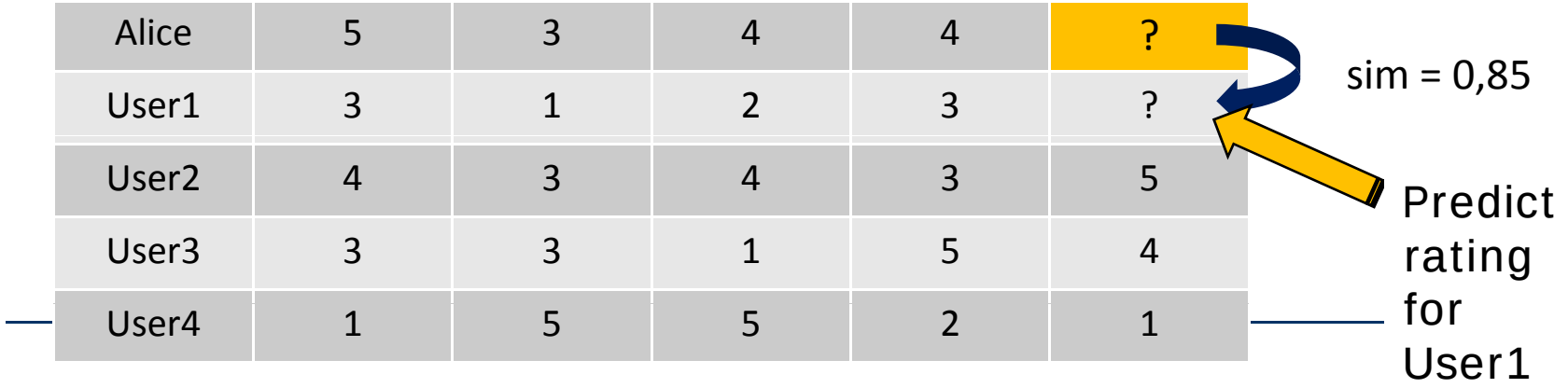
Data sparsity problems

- **Cold start problem**
  - How to recommend new items? What do recommend to new users?
- **Straightforward approaches**
  - Ask/force users to rate a set of items
  - Use another method (e.g., content-based, demographic or simply non-personalized) in the initial phase
- **Alternatives**
  - Use better algorithms (beyond nearest-neighbor approaches)
  - Example:
    - In nearest-neighbor approaches, the set of sufficiently similar neighbors might be to small to make good predictions
    - Assume "transitivity" of neighborhoods

Example algorithms for sparse datasets

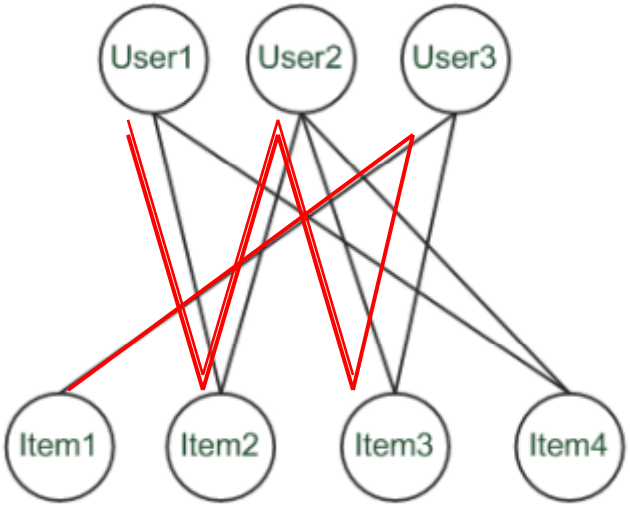
- **Recursive CF**
  - Assume there is a very close neighbor n of u who however has not rated the target item i yet.
  - Idea:
    - Apply CF-method recursively and predict a rating for item i for the neighbor
    - Use this predicted rating instead of the rating of a more distant direct neighbor

	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



Graph-based methods

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



More model-based approaches

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

Matrix factorization models

- **Latent Semantic Indexing**
  - developed in the information retrieval field; aims at detection of hidden "factors" (topics) of a document and the reduction of the dimensionality
  - allows retrieval of documents that do not contain query keywords
  - based on Singular Value Decomposition (SVD)
- **SVD-based recommendation**
  - decompose matrix / find factors
    - factors can be genre, actors but also non-understandable ones
  - only retain the n=20 to 100 most important factors
    - can also help to remove noise in the data
  - make recommendation in the lower-dimensional space
    - e.g., use nearest neighbors
- **Heavily used in Netflix prize competition; specific methods proposed**

Association rule mining

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

    "If a customer purchases baby-food then he also buys diapers in 70% of the cases"
- **Association rule mining algorithms**
  - can detect rules of the form X => Y (e.g., baby-food => diapers) from a set of sales transactions
  - measure of quality: support, confidence
    - used as a threshold to cut off unimportant rules

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

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

Recommendation based on Association Rule Mining

- **Simplest approach**
  - transform 5-point ratings into binary ratings (1 = above user average)
- **Mine rules such as**
  - Item1 => Item5
    - support (2/4), confidence (4/4) (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)
  - Compute union of Y's not already bought by Alice
  - Sort the products based on the rules' confidence values
- **Different variations possible**
  - dislike statements, user associations ..

	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

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

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

  - Assumption: Ratings are independent (?)

Calculation of probabilities in simplistic approach

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

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

- **Does not work in practice …**
  - Zeros (smoothing required), computationally expensive, …
  - like/dislike simplification possible

Practical probabilistic approaches

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

Collaborative Filtering Issues

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

Content-based recommendation

Content-based recommendation

- **While CF – methods do not require any information about the items,**
  - it might be reasonable to exploit such information; and
  - recommend fantasy novels to people who liked fantasy novels in the past
- **What do we need:**
  - some information about the available items such as the genre ("content")
  - some sort of *user profile* describing what the user likes (the preferences)
- **The task:**
  - learn user preferences
  - locate/recommend items that are "similar" to the user preferences

What is the "content"?

- **The genre is actually not part of the content of a book**
- **Most CB-recommendation methods originate from Information Retrieval (IR) field:**
  - goal is to find and rank interesting text documents (news articles, web pages)
  - the item descriptions are usually automatically extracted (important words)
- **Fuzzy border between content-based and "knowledge-based" RS**
- **Here:**
  - classical IR – based methods based on keywords
  - no means-ends recommendation knowledge involved
  - User profile (preferences) are rather learned than explicitly elicited

Content representation and item similarities

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

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

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

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

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

- **Simple keyword representation has its problems**
  - in particular when automatically extracted as
    - not every word has similar importance
    - longer documents have a higher chance to have an overlap with the user profile
- **Standard measure: TF-IDF**
  - Encodes text documents in multi-dimensional Euclidian space
    - weighted term vector
  - TF: Measures, how often a term appears (density in a document)
    - assuming that important terms appear more often
    - normalization has to be done in order to take document length into account
  - IDF: Aims to reduce the weight of terms that appear in all documents
  - Given a keyword i and a document j

    TF-IDF (i,j) = TF(i,j) * IDF(i)

Example TF-IDF representation

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

Figure taken from http://informationretrieval.org

More on the vector space model

- **Vectors are usually long and sparse**
- **Improvements**
  - remove stop words ("a", "the", ..)
  - use stemming
  - size cut-offs (only use top n most representative words, e.g. around 100)
  - use lexical knowledge, use more elaborate methods for feature selection
  - detection of phrases as terms (such as 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.."
- **Usual similarity metric to compare vectors: Cosine similarity (angle)**

Recommending items

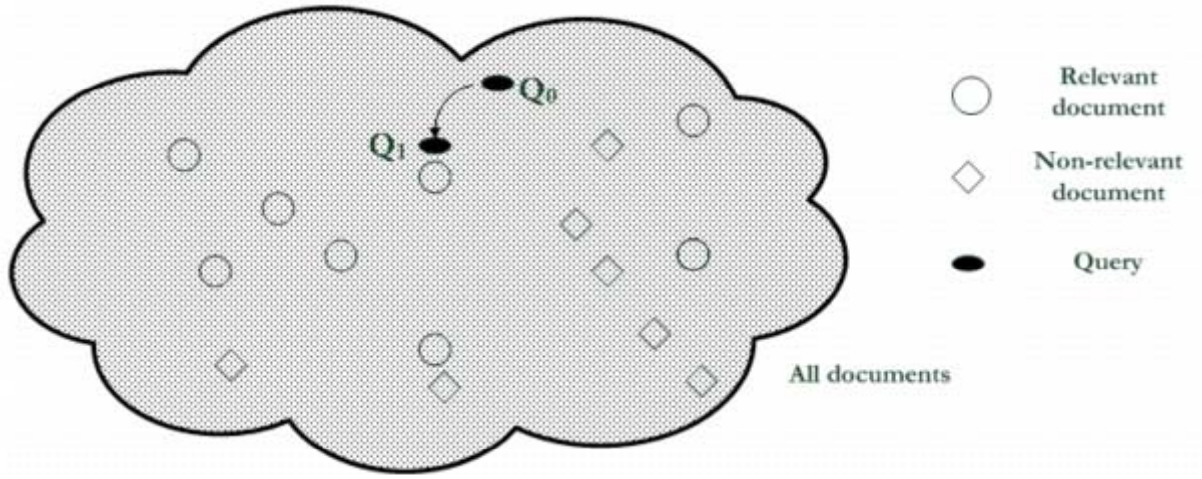
- **Simple method: nearest neighbors**
  - Given a set of documents D already rated by the user (like/dislike)
    - Find the n nearest neighbors of an not-yet-seen item i in D
    - Take these ratings to predict a rating/vote for i
    - (Variations: neighborhood size, lower/upper similarity thresholds..)
  - Good to model short-term interests / follow-up stories
  - Used in combination with method to model long-term preferences
- **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^+ and D^-
- α, β, γ used to fine-tune the feedback

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

- often only positive feedback is used



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	**?**

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

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
- **Other linear classification algorithms (machine learning) can be used**
  - Support Vector Machines, ..
- **Use other information retrieval methods (used by search engines..)**

Limitations of content-based recommendation methods

- **Keywords alone may not be sufficient to judge quality/relevance of a document or web page**
  - up-to-dateness, usability, aesthetics, writing style
  - content may also be limited / too short
  - content may not be automatically extractable (multimedia)
- **Ramp-up phase required**
  - Some training data is still required
  - Web 2.0: Use other sources to learn the user preferences
- **Overspecialization**
  - Algorithms tend to propose "more of the same"
  - Or: too similar news items

Knowledge-Based Recommender Systems

Knowledge-Based Recommendation I

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

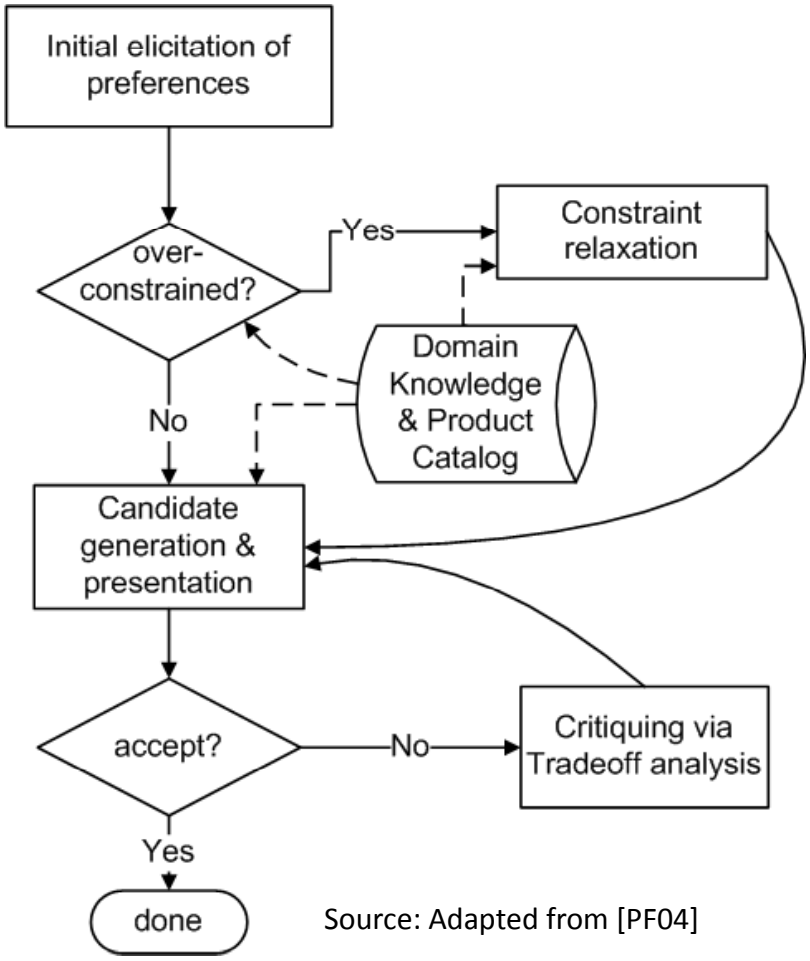
- **Explicit domain knowledge**
  - Requirements elicitation from domain experts
  - System mimics the behaviour of experienced sales assistant
  - Best-practice sales interactions
  - Can guarantee “correct” recommendations (determinism)

Knowledge-Based Recommendation II

- **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
  - Interactive constraint/preference acquisition from users
    - Critiques or local preferences
  - Declarative knowledge-base (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

Conversation strategies



Source: Adapted from [PF04]

- **Initial needs- or feature-based preference elicitation**
  - Candidate generation
  - Identify maximally succeeding subsets (XSS) to avoid empty result sets
- **Repeated interaction cycles**
  - Critiques (more of A and less B)
  - Revision of preference models

Constraint-based recommendation I

- **Knowledge base**
  - Usually mediates between user model and item properties
  - 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**
  - Fulfilling set of applicable constraints
  - Applicability of constraints depends on current user model
  - Explanations – transparent line of reasoning

Constraint-based recommendation II

Several different recommendation tasks:

- **Find a set of user requirements such that a subset of items fulfills all constraints**
  - Ask user which requirements should be relaxed/modified such that some items exist that do not violate any constraint (e.g. Trip@dvice [MRB04])

- **Find a subset of items that satisfy the maximum set of weighted constraints**
  - Similar to find a maximally succeeding subquery (XSS) [McSherry05]
  - All proposed items have to fulfill the same set of constraints (e.g. [FFJ+06])
  - Compute relaxations based on predetermined weights

- **Rank items according to weights of satisfied soft constraints**
  - Rank items based on the ratio of fulfilled constraints
  - Does not require additional ranking scheme

Example

Knowledge Base:

Weight	LHS	RHS
C1: 25	TRUE	Brand = Brand pref.
C2: 20	Motives = *Landscape*	Low. foc. Length =< *28*
C3: 15	TRUE	Price =< Max. cost

Current user:

User model (requirements)	
Motives	*Landscape*
Brand preference	*Canon*
Max. cost	*350 EUR*

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*

Ask user

- **Do you want to relax your brand preference?**
  - Accept *Panasonic* instead of *Canon* brand
- **Or is photographing landscapes with a wide-angle lens and maximum cost less important?**
  - Lower focal length > 28mm and Price > 350 EUR
- **Computation of minimal revisions of requirements**
  - Eventually guided by some predefined weights or past community behavior

Find XSS

- **Two maximally succeeding subqueries**
  - XSS = {{C1},{C2, C3}}
  - Selection can be based on constraints‘ weights
  - Relax C1 and recommend *Lumix*

Compute scores

- Applicable: LHS(c) is satisfied by user model, i.e. {C1,C2,C3}
- Satisfied: not applicable or RHS(c) is satisfied by catalogue item, i.e. {C1} for Powershot and {C2,C3} for Lumix

$$rec_\gamma(i,u,\Gamma) = \begin{cases} score(i,u,\Gamma) & : \forall c \in C_{hard}\ sat(c) \\ 0 & : \text{else} \end{cases}$$

- Only items that satisfy all hard constraints receive positive score (fulfilled for both)

$$score(i,u,\Gamma) = \frac{1}{\sum_{\substack{\forall c \in C \\ app(c)}} pen(c)} \times \sum_{\substack{\forall c \in C \\ app(c) \wedge sat(c)}} pen(c)$$

- Ratio of penalty values of satisfied constraints
  - **Ranked #1: Lumix 35/60**
  - **Ranked #2: Panasonic: 25/60**
  - Cutoff recommendation list after *n* items

Constraint-based recommendation III

- **More variants of constraint representation:**
  - Interactive acquisition of solution preferences
    - E.g. user can ask „less price“ for *Lumix*
    - To explore cheaper variants of current proposal
    - Max. cost of 350 EUR for *Canon* brand initially specified, higher price sensitivity for *Panasonic* brand?
    - Aging/Outdating of „older“ preferences
    - Construction of decision model / trade-off analysis
  - Disjunctive RHS
    - IF location requ.= *nearby* THEN location = *Ktn* OR location = *Stmk*
    - E.g. The thermal spa should be located either in Carinthia or Styria

Constraint-based recommendation IV

- **More variants of recommendation task**
  - Find „diverse“ sets of items
    - Notion of similarity/dissimilarity
    - Idea that users navigate a product space
    - If recommendations are more diverse than users can navigate via critiques on recommended „entry points“ more efficiently (less steps of interaction)
  - Bundling of recommendations
    - Find item bundles that match together according to some knowledge
      - E.g. travel packages, skin care treatments or financial portfolios
      - RS for different item categories, CSP restricts configuring of bundles

Constraint-based recommendation V

- **Find optimal sequence of conversational moves**
  - „Recommendation is less about optimal algorithms, but more about the interaction experience“ [MR09]
  - Asking for requirements, proposing items (that can be critiqued) or showing explanatory texts are all conversational moves
  - Interaction process towards preference elicitation and maybe also user persuasion
  - Exploit current state of the system as well as community information

Limitations of knowledge-based recommendation methods

- **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
  - E.g. asymmetric dominance effects and Decoy items

Hybridization Strategies

Hybrid recommender systems

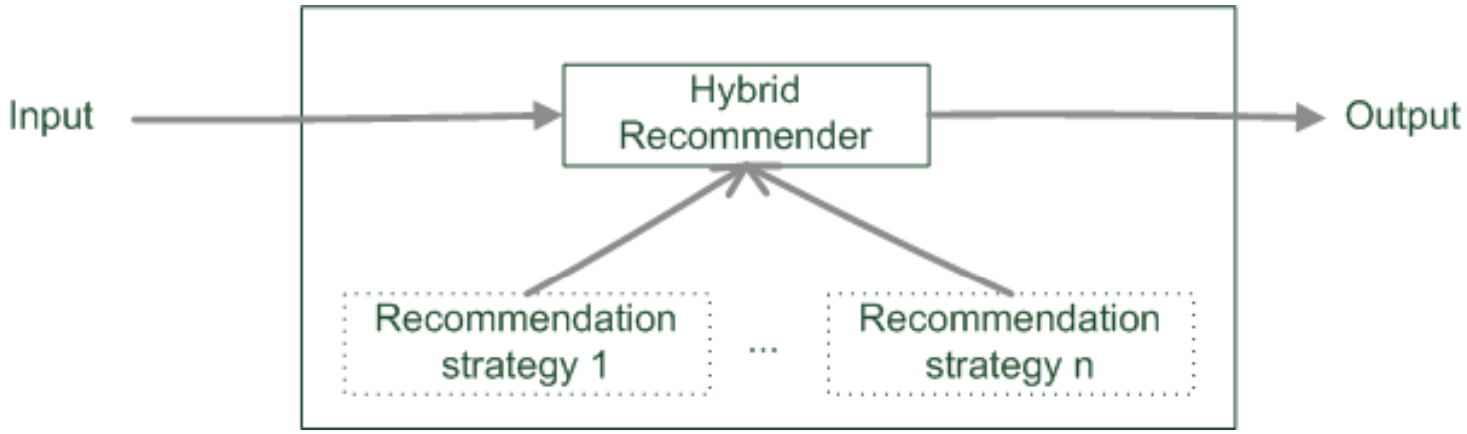
- **All three base techniques are naturally incorporated by a good sales assistance (at different stages of the sales act) but have their shortcomings**

- **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**
  - Parallel use of several systems
  - Monolithic exploiting different features
  - 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

- **Combination of several knowledge sources**
  - E.g.: Ratings and user demographics or explicit requirements and needs used for similarity computation

- **'Hybrid' content features:**
  - Social features: Movies liked by user
  - Content features: Comedies liked by user, dramas liked by user
  - Hybrid features: user likes many movies that are comedies, …

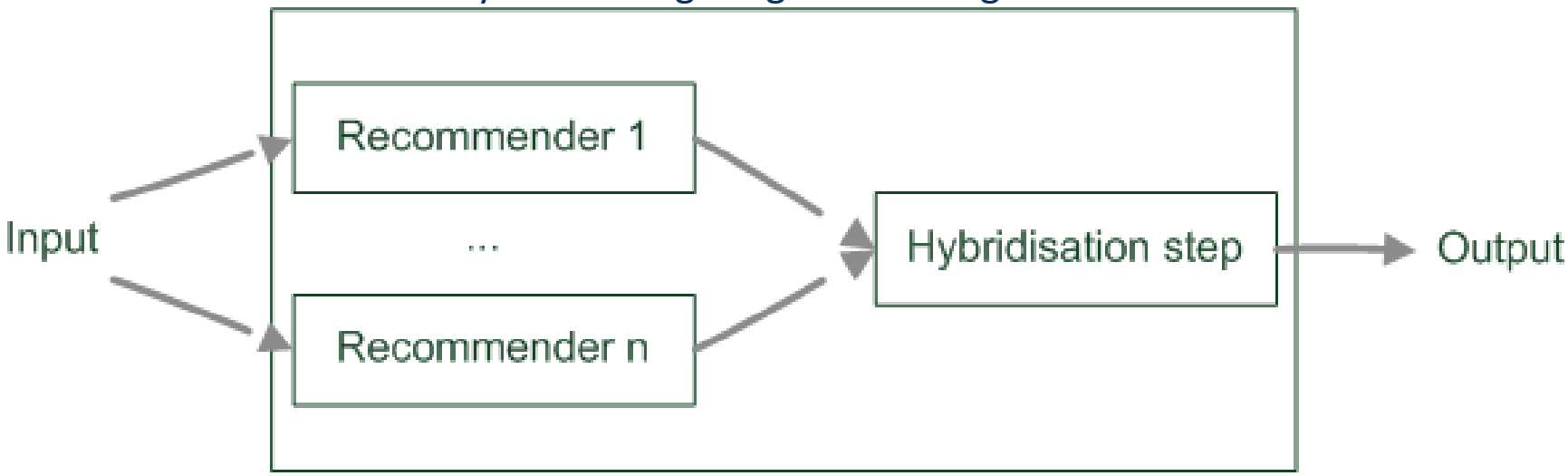
  - "*the common knowledge engineering effort that involves inventing good features to enable successful learning*" [BHC98]

Monolithic hybridization designs: Feature augmentation

- **Content-boosted collaborative filtering [MMN02]**
  - 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 [TMA+04]**
  - Citations interpreted as collaborative recommendations

Parallelized hybridization design

- **Output of several existing implementations combined**
- **Least invasive design**
- **Some weighting or voting scheme**
  - Weights can be learned dynamically
  - Extreme case of dynamic weighting is switching



Parallelized hybridization 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)

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

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	

Parallelized hybridization 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 hybridization design: Switching

- **Requires an oracle that decides on recommender**

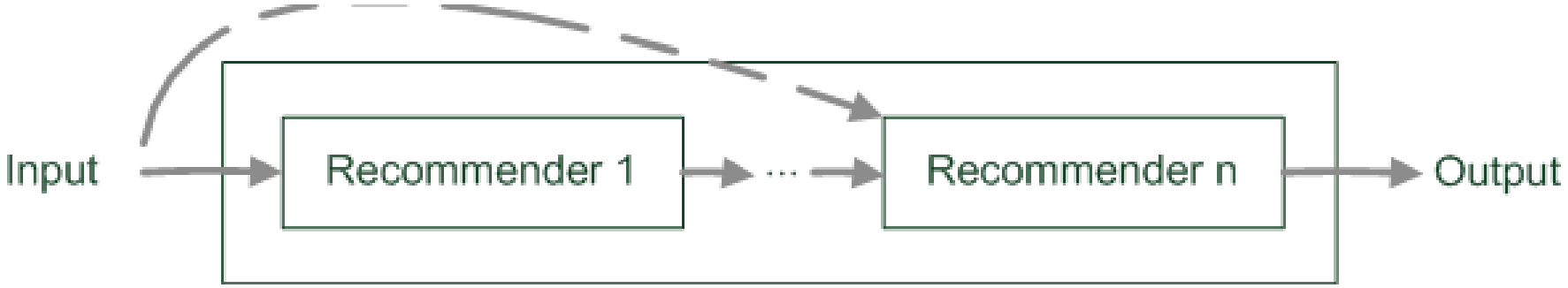
$$\exists_1 k : 1 \ldots n \;\; rec_{switching}(u, i) = rec_k(u, i)$$

- **Special case of dynamic weights (all except one Beta is 0)**

- **Example:**
  - Ordering on recommenders and switch based on some quality criteria
    - E.g. if too few ratings in the system use knowledge-based, else collaborative
  - More complex conditions based on contextual parameters, apply classification techniques

Pipelined hybridization designs

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



Pipelined hybridization designs: Cascade

- Successor's recommendations are restricted by predecessor

$$rec_{cascade}(u,i) = rec_n(u,i)$$

- Where forall k > 1

$$rec_k(u,i) = \begin{cases} rec_k(u,i) & : rec_{k-1}(u,i) \neq 0 \\ 0 & : \text{else} \end{cases}$$

- Subsequent recommender may not introduce additional items
- Thus produces very precise results

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 Delta built by predecessor**

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

- **Examples:**
  - Fab:
    - Online news domain
    - CB recommender builds user models based on weighted term vectors
    - CF identifies similar peers based on these user models but makes recommendations based on ratings
  - Collaborative constraint-based meta-level RS
    - Collaborative filtering learns a constraint base
    - Knowledge-based RS computes recommendations

Limitations of hybridization strategies

- **Only few works that compare strategies from the meta-perspective**
  - Like for instance, [Burke02]
  - 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-in 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, context and meta-features

Evaluation of Recommender Systems

Evaluating Recommender Systems

- **A myriad of techniques has been proposed, but**
  - Which one is best in a given application domain?
  - What are the success factors of different techniques?
  - Comparative analysis based on an optimality criterion?

- **Research questions are:**
  - Is a RS efficient with respect to a specific criteria like accuracy, user satisfaction, response time, serendipity, online conversion, ramp-up efforts, ….
  - Do customers like/buy recommended items?
  - Do customers buy items they otherwise would have not?
  - Are they satisfied with a recommendation after purchase?

Purpose and success criteria I

- **Different perspectives/aspects**
  - Depends on domain and purpose
  - No wholistic evaluation scenario exists

- **Retrieval perspective**
  - Reduce search costs
  - Provide ‚correct' proposals
  - Users know in advance what they want

- **Recommendation perspective**
  - Serendipity
  - Users did not know about existence

Purpose and success criteria II

- **Prediction perspective**
  - Predict to what degree users like an item
  - Most popular evaluation scenario in research
- **Interaction perspective**
  - Give users a ‚good feeling‘
  - Educate users about the product domain
  - Persuade users as an intentional planned effect!?
- **Finally, conversion perspective**
  - Commercial situations
  - Increase ‚hit‘, ‚clickthru‘, ‚lookers to bookers‘ rates
  - Optimize sales margins and profit

Empirical research

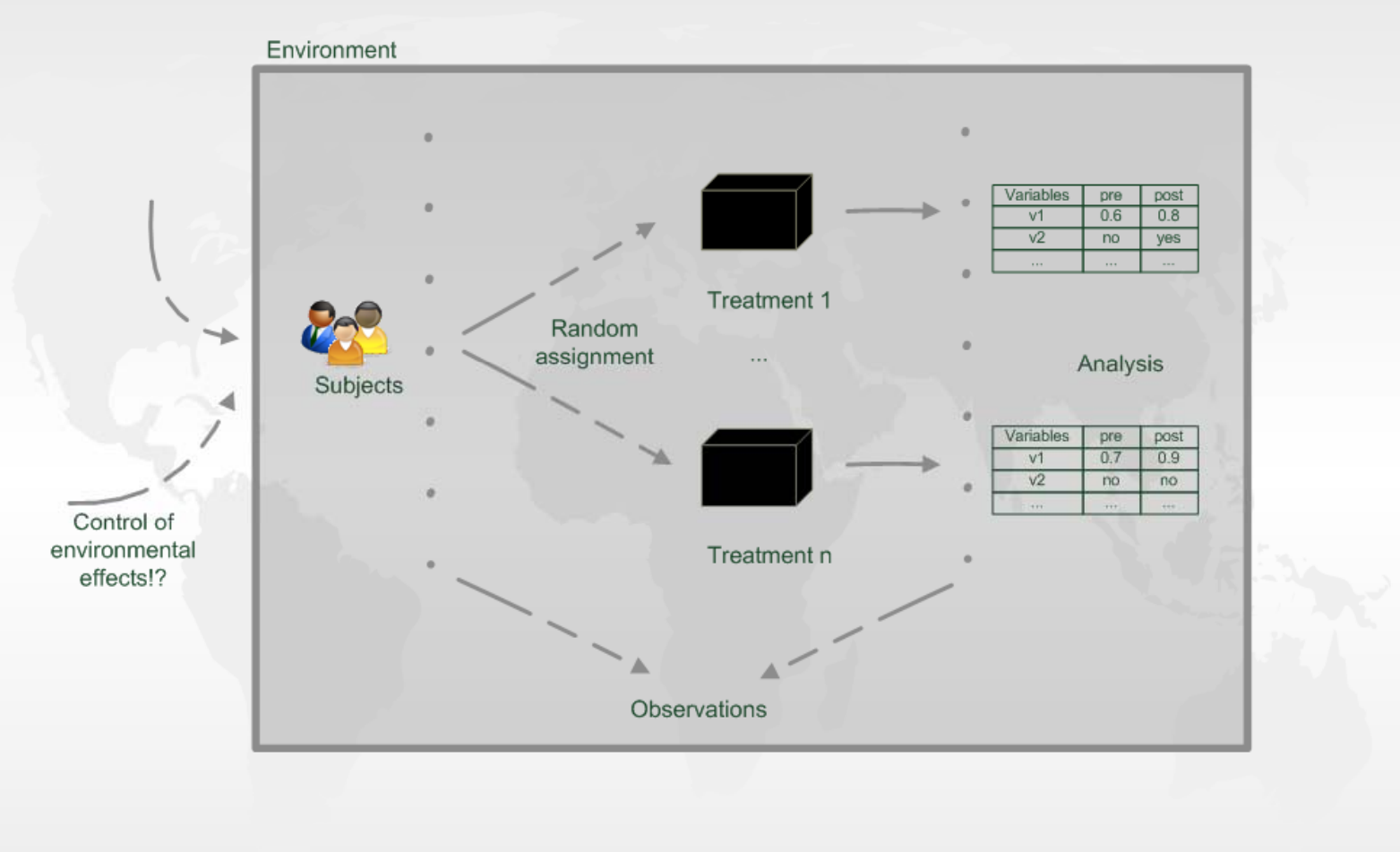
- **Characterizing dimensions:**
  - Who is the **subject** that is in the focus of research?
  - What **research method**s are applied?
  - In which **setting** does the research take place?

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

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, recommended items, …
    - Categories of manipulated variable(s): content-based RS, collaborative RS

Experiment designs



Metrics: MAE and RMSE

- **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**

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

  - *RMSE* is used, for example, in the evaluation of the NetFlix Prize

Metrics: Precision and Recall I

- **Precision and Recall, two standard measures from Information Retrieval, are two of the most basic methods for evaluating recommender systems**

- **E.g. Consider the movie predictions made by a simplified recommender that classes movies as good or bad**
  - They can be split into four groups:

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

Metrics: Precision and Recall II

- **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

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

Metrics: Precision and Recall III

Offline experimentation	Online experimentation
Historic session	Live interaction
Ratings, transactions	Ratings, feedback
Unrated items unknown, but interpreted as dislikes	Dislikes of not recommended items unknown
False positives might be wrong	False negatives cannot be determined
Better for estimating Recall	Better for estimating Precision

Metrics: Rank Score

- **Rank Score extends the recall metric to take the positions of correct items in a ranked list into account**
  - Particularly important in recommender systems as lower ranked items may be overlooked by users
- **Rank Score is defined as the ratio of the Rank Score of the correct items to best theoretical Rank Score achievable for the user, i.e.**

$$rankscore = \frac{rankscore_p}{rankscore_{max}}$$

$$rankscore_p = \sum_{i \in h} 2^{-\frac{rank(i)-1}{\alpha}}$$

$$rankscore_{max} = \sum_{i=1}^{|T|} 2^{-\frac{i-1}{\alpha}}$$

Where:
- *h* is the set of correctly recommended items, i.e. hits
- *rank* returns the position (rank) of an item
- *T* is the set of all items of interest
- *α* is the *ranking half life*

Offline experimentation

- Netflix competition
  - Web-based movie rental
  - Prize of $1,000,000 for accuracy improvement 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
    - Test set – used by Netflix to determine winner

Learnings

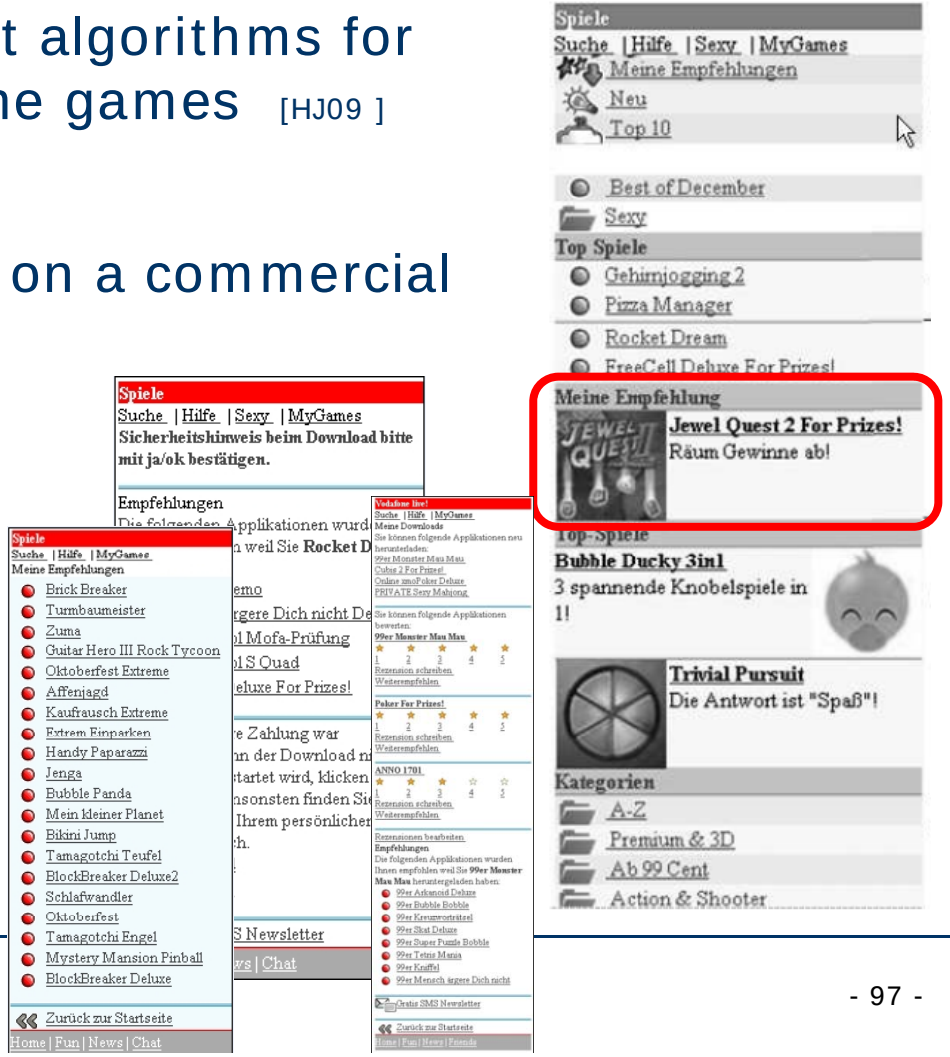
- **Winning team combined more than 100 different predictors**
  - Small group of controversial movies responsible for high share of error rate
  - E.g. singular value decomposition, a technique for deriving the underlying factors that cause viewers to like or dislike movies, can be used to find connections between movies
  - Interestingly, most teams used similar prediction techniques

- **Very complex and specialized models**
  - Switching of model parameters based on user/session features
  - Number of rated items
  - Number of rated items on particular day

- **Content features added noise**

Online experimentation

- Effectiveness of different algorithms for recommending cell phone games [HJ09]
- Involved 150,000 users on a commercial mobile internet portal
- A/B testing
- Results:
  - See case study on the mobile Internet



Non-experimental research

- **Quasi-experiments**
  - Lack random assignments of units to different treatments

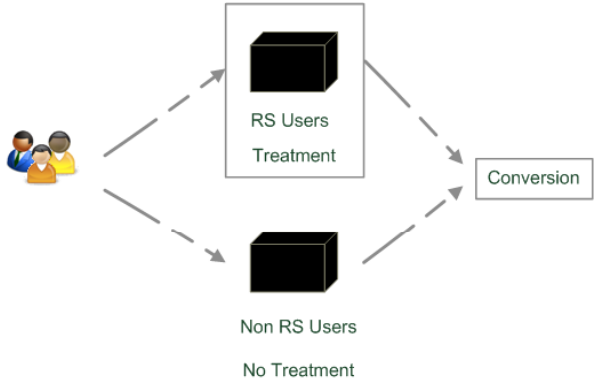
- **Non-experimental / observational research**
  - Surveys / Questionnaires
  - Longitudinal research
    - Observations over long period of time
    - E.g. Customer life-time value, returning customers
  - Case studies
  - Focus group
    - Interviews
    - Think aloud protocols

Quasi-experimental

- **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

- **Delgado and Davidson evaluated the effectiveness of the recommender 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, such as**
  - Are users of the recommender systems more likely convert?
  - Does the recommender system itself cause users to convert?

- **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

What is popular?

- **Evaluations on historical datasets measuring accuracy**

- **Most popular datasets**
  - Movies (MovieLens, EachMovie, Netflix)
  - Web 2.0 platforms (tags, music, papers, …)

- **Most popular measures for accuracy**
  - Precision/Recall
    - If item is either relevant or not
  - MAE (Mean Absolute Error), RMSE (Root Mean Squared Error)
    - If we want to predict ratings (on a Likert scale)

What is popular?

- **Evaluation designs ACM TOIS 2004-2008**
  - In total 12 articles on RS
  - 50% movie domain
  - 75% offline experimentation
  - 2 user experiments under lab conditions
  - 1 qualitative research

- **Availability of data heavily biases what is done**
  - Many „tag recommenders“ proposed recently
  - Tenor at RecSys’09 to foster live experiments
  - Public infrastructures to enable A/B tests

Agenda

- **What are recommender systems for?**
  - Introduction
- **How do they work?**
  - Collaborative Filtering
  - Content-based Filtering
  - Knowledge-Based Recommendations
  - Hybridization Strategies
- **How to measure their success?**
  - Evaluation techniques
  - Case study on the mobile Internet
- **Selected recent topics**
  - Attacks on CF Recommender Systems
  - Recommender Systems in the Social Web
- **What to expect?**

Recommendations on the Mobile Internet

Case studies and evaluations in RS research

- **The MovieLens data set, others**
  - Focus on improving the Mean Absolute Error …
- **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

Experimental setup: the game download platform

- **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"

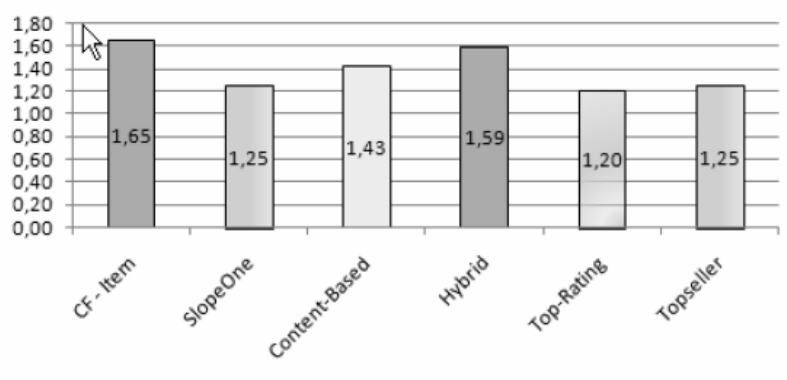
Experiment setup (ctd.)

- **6 recommendation algorithms, 1 control group (A/B test)**
  - 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
- **Hypothesis (H1 – H4)**
  - H1: Pers. recommendations **stimulate more users** to **view items**
  - H2: Person. recommendations **turn more visitors into buyers**
  - H3: Pers. recommendations stimulate **individual users** to **view more items**
  - H3: Pers. recommendations stimulate **individual users** to **buy more items**

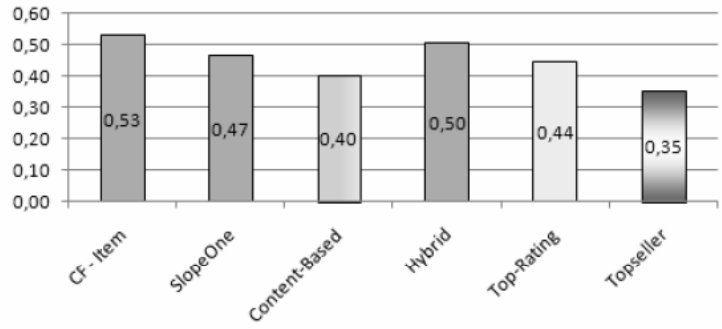
Measurements

- **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 ratings**
  - Item view, item purchase, explicit ratings

Measurement: "My Recommendations" - section



Item views/ customer



Purchases / customer

- **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

– **Conversion rates: No strong effects**

Measurement: My Recommendations sales increase

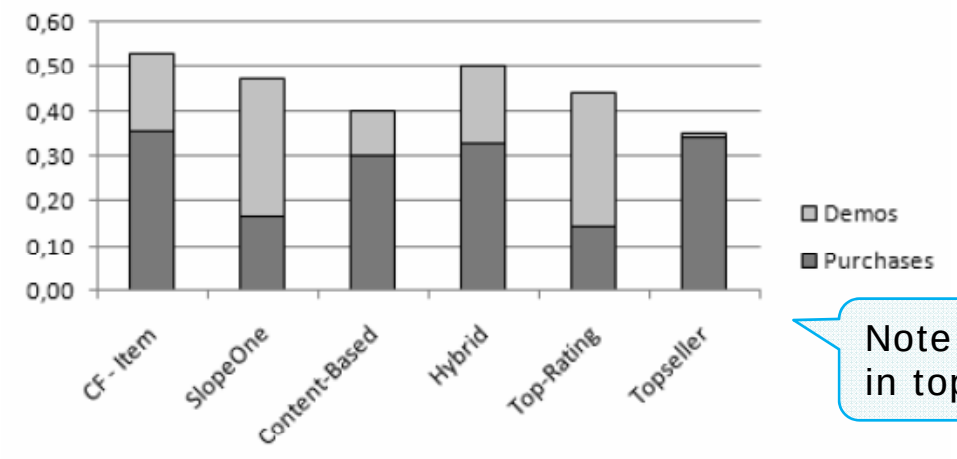
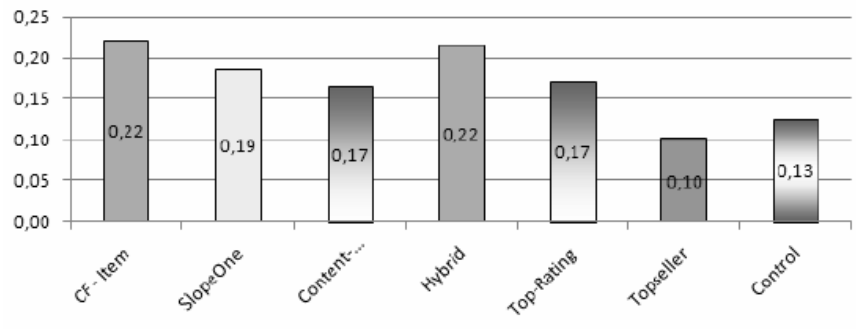


Figure shows purchases per visitor rate

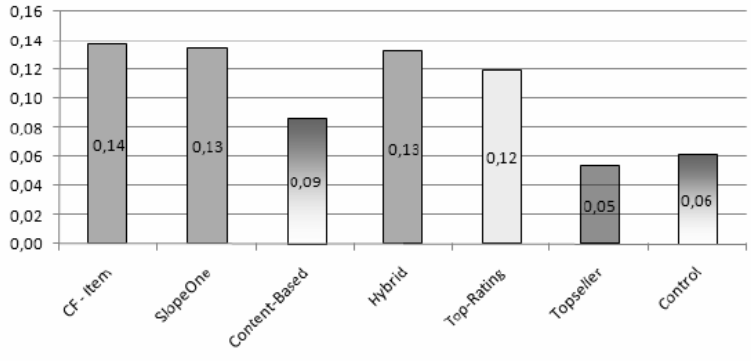
Note: Only 2 demos in top 30 downloads

- **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
- **Note: Rating possible only after download**

Measurement: Post-sales recommendations



Item views / visitor

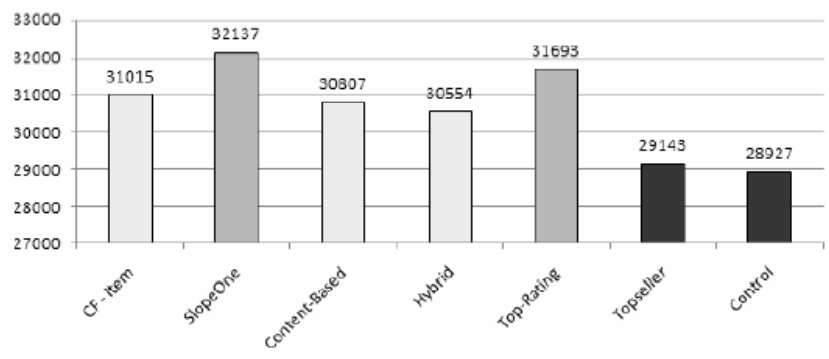


Purchases / visitor

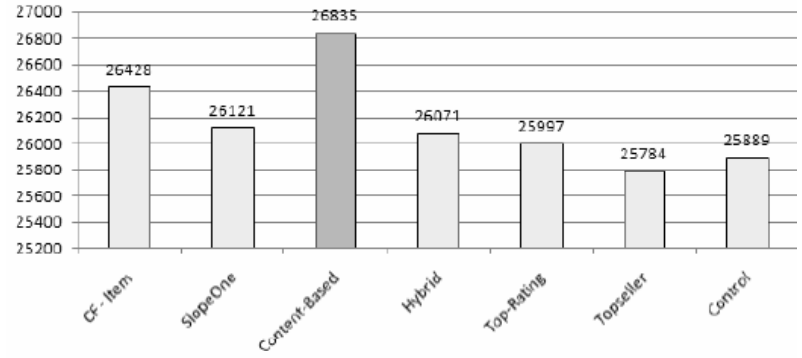
- **Findings**
  - Recommending "more-of-the-same", top sellers or simply new items does not work well
  - Top-Rating and SlopeOne nearly exclusively stimulate demo downloads (Not shown)
  - Top-Seller und control group sell no demos

Measurement: Overall effects

- **Overall number of downloads (free + non-free games)**



- **Pay games only**



Notes:

In-category measurements not shown here.

Content-based method outperforms others in different categories

(half price, new games, erotic games)

Effect: 3.2 to 3.6% sales increase!

Final observations: Recommendations on the Mobile Internet

- **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
  - Insufficient coverage for standard CF methods
- **Implicit ratings**
  - Also count item views and item purchases
  - Increase the coverage of CF algorithms
  - MAE however not a suitable measure anymore for comparing algorithms
- **Summary**
  - Significant sales increase can be reached! (max. 1% in past with other activities)
  - More studies needed, Value of MAE measure …
  - Recommendation in navigational context

Agenda

- **What are recommender systems for?**
  - Introduction
- **How do they work?**
  - Collaborative Filtering
  - Content-based Filtering
  - Knowledge-Based Recommendations
  - Hybridization Strategies
- **How to measure their success?**
  - Evaluation techniques
  - Case study on the mobile Internet
- **Selected recent topics**
  - Attacks on CF Recommender Systems
  - Recommender Systems in the Social Web
- **What to expect?**

Attacks on CF Recommender Systems

Introduction / Background

- **(Monetary) value of being in recommendation lists**
  - Individuals may be interested to push some items by manipulating the recommender system
  - Individuals might be interested to decrease the rank of other items
  - Some simply might may want to sabotage the system ..
- **Manipulation of the "Internet opinion"**
  - Not a new issue ..
- **A simple strategy?**
  - (Automatically) create numerous fake accounts / profiles
  - Issue high or low ratings to the "target item"

    ==> Will not work for neighbor-based recommenders

    ==> More elaborate attack models required

    == > Goal is to insert profiles that will appear in neighborhood of many

Characterization of profile insertion attacks

- **Push / Nuke**
  - Not the same effects observed
- **Cost**
  - How costly is it to make an attack?
  - How many profiles have to be inserted?
  - Is knowledge about the ratings matrix required?
    - usually it is not public, but estimates can be made
- **Algorithm dependability**
  - Is the attack designed for a particular recommendation algorithm?
- **Detectability**
  - How easy is it to detect the attack

The Random Attack

- **General scheme of an attack profile**

Item1	...	ItemK	...	ItemL	...	ItemN	Target
r_1	...	r_k	...	r_l	...	r_n	X
selected items			filler items		unrated items		

  - Attack models mainly differ in the way the profile sections are filled

- **Random attack model**
  - High/low ratings for target items
  - Take random values for filler items
    - Typical distribution of ratings is known, e.g., for the movie domain (Average 3.6, standard deviation around 1.1)
  - Limited effect compared with more advanced models

More attack models

- **The Average Attack**
  - use the individual item's rating average for the filler items
  - intuitively, there should be more neighbors
  - additional cost involved: find out the average rating of an item
  - more effective than Random Attack in user-based CF
- **By the way: what does effective mean?**
  - Possible metrics to measure the introduced bias
  - Robustness
    - deviation in general accuracy of algorithm
  - Stability
    - change in prediction for a target item (before/after attack)
  - In addition: rank metrics
    - How often does an item appear in Top-N lists (before/after)

More examples of attack models

- **Bandwagon Attack**
  - Exploits additional information about the community ratings
  - Simple idea:
    - Add profiles that contain high ratings for "blockbusters" (in the selected items); use random values for the filler items
    - Will intuitively lead to more neighbors
- **Segment Attack**
  - Find items that are similar to the target item, i.e., are probably liked by the same group of people (e.g., other fantasy novels)
  - Inject profiles that have high ratings for fantasy novels and random or low ratings for other genres
  - Thus, item will be pushed within the relevant community

Effectiveness analysis

- **In general**
  - Effect depends mainly on the attack size (number of fake profiles inserted)
- **User-based recommenders:**
  - **Bandwagon / Average Attack:** Bias shift of 1.5 points on a 5-point scale at 3% attack size (3% of profiles are faked after the attack)
  - Average Attack slightly better but requires more knowledge
  - 1.5 points shift is significant; 3% attack size however means inserting e.g., 30.000 profiles into one-million rating database …
- **Item-based recommenders**
  - Far more stable; only 0.15 points prediction shift achieved
  - Exception: Segment attack successful (was designed for item-based method)
  - Hybrid recommenders and other model-based algorithms cannot be easily biased (with the described/known attack models)

Counter measures

- **Use model-based or hybrid algorithms**
- **Increase profile injection costs**
  - Captchas
  - Low-cost manual insertion …
- **Use statistical attack detection methods**
  - detect groups of users who collaborate to push/nuke items
  - monitor development of ratings for an item
    - changes in average rating
    - changes in rating entropy
    - time-dependent metrics (bulk ratings)
  - use machine-learning methods to discriminate real from fake profiles

Discussion

- **Not discussed here: Privacy-ensuring methods**
  - Distributed collaborative filtering, data perturbation
- **Research on attacks**
  - Vulnerability of some existing methods shown
  - Specially-designed attack models may also exist for up-to-now rather stable methods
  - Incorporation of more knowledge-sources /hybridization may help
- **Practical aspects**
  - No public information on large-scale real-world attack available
  - Attack sizes are still relatively high
  - More research and industry-collaboration required

Agenda

- **What are recommender systems for?**
  - Introduction
- **How do they work?**
  - Collaborative Filtering
  - Content-based Filtering
  - Knowledge-Based Recommendations
  - Hybridization Strategies
- **How to measure their success?**
  - Evaluation techniques
  - Case study on the mobile Internet
- **Selected recent topics**
  - Attacks on CF Recommender Systems
  - Recommender Systems in the Social Web
- **What to expect?**

Recommender Systems in the Social Web

RS and the Social Web

- **The Web 2.0 / Social Web**
  - Facebook, Twitter, Flickr, …
  - People actively contribute information and participate in social networks
- **Impact on recommender systems**
  - More information about user's and items available
    - demographic information about users
    - friendship relationships
    - tags on resources
  - New application fields for RS technology
    - Recommend friends, resources (pictures, videos), or even tags to users

  ==> Requires the development of new algorithms

  ==> Currently, lots of papers published on the topic

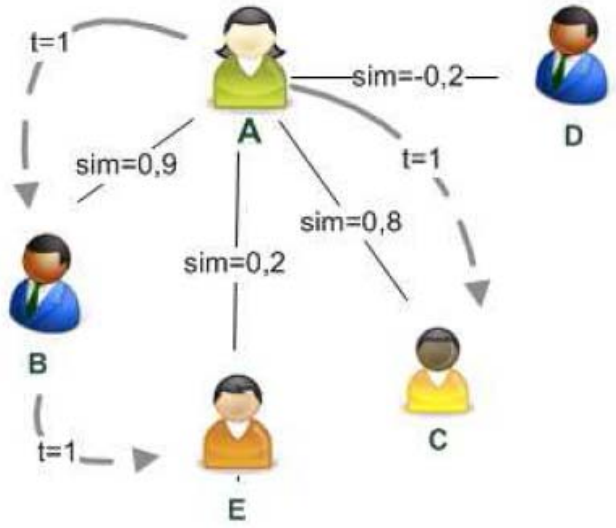
Trust-aware recommender systems

- **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

- **Exploiting trust information in RS**
  - to improve accuracy (neighborhood selection)
  - to increase coverage
  - could be used to make RS robust against attacks

Example system: TARS (Massa & Avensani)

- **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

Aspects of 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

Tag-based CF and item retrieval

- **Item retrieval in Web 2.0 applications**
  - often based on overlap of query terms and item tags
  - insufficient for retrieving the "long tail" of items
    - users may use different terms in their annotations
      - think of possible tags of a car: "Volkswagen", "beetle", "red", "cool"…
- **One approach: Social Ranking**
  - use CF methods to retrieve ranked list of items for given query
    - compute user and tag similarities (e.g., based on co-occurrence)
  - two-phase retrieval
    - extend user query with similar tags (improves coverage)
    - rank items based on
      - relevance of tags to the query
      - similarity of taggers to the current user
  - leads to measurably better coverage and long-tail retrieval

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

Agenda

- **What are recommender systems for?**
  - Introduction
- **How do they work?**
  - Collaborative Filtering
  - Content-based Filtering
  - Knowledge-Based Recommendations
  - Hybridization Strategies
- **How to measure their success?**
  - Evaluation techniques
  - Case study on the mobile Internet
- **Selected recent topics**
  - Attacks on CF Recommender Systems
  - Recommender Systems in the Social Web
- **What to expect?**

Outlook hypotheses

„RS research will become much more diverse“

- **Less focus on explicit ratings**
  - But various forms of feedback mechanisms and knowledge
  - Social and Semantic Web, automated knowledge extraction
  - Context-awareness (beyond geographical positions)
- **Less focus on algorithms**
  - But also on interfaces and interaction processes
  - Explaining and trust-building
  - Persuasive aspects
- **Less focus on offline experimentation**
  - But live experiments, real-world case studies, …
- **More focus on causal relationships**
  - When, where and how to recommend?
  - Consumer / Sales psychology
  - Consumer decision making theories

Thank you for your attention!

Questions?

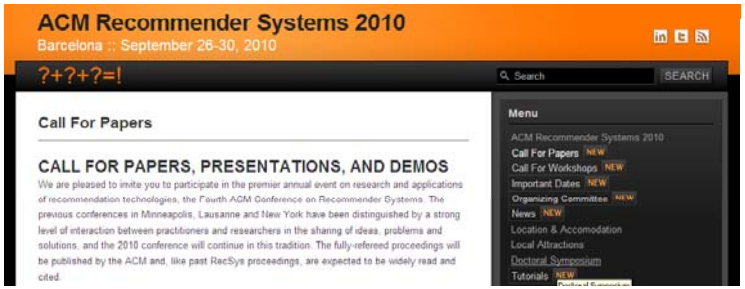
Questions?

Questions?

Dietmar Jannach
e-Services Research Group
Department of Computer Science
TU Dortmund, Germany
M: dietmar.jannach@tu-dortmund.de
P: +49 231 755 7272

Markus Zanker
Intelligent Systems and Business Informatics
Institute of Applied Informatics
University Klagenfurt, Austria
M: markus.zanker@uni-klu.ac.at
P: +43 463 2700 3753

http://recsys.acm.org



http://www.recommenderbook.net

Recommender Systems – An Introduction by

Dietmar Jannach, Markus Zanker, Alexander Felfernig and Gerhard Friedrich
Cambridge University Press, to appear 2010/11

References

[AT05] Adomavicius & Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE TKDE, 17(6), 2005, pp.734-749.

[BHC98] Basu, Hirsh & Cohen. Recommendation as classification: using social and content-based information in recommendation, AAAI98, pp. 714-720.

[Burke02] Burke. Hybrid Recommender Systems: Survey and Experiments. UMUAI 12(4), 2002, 331-370.

[FFJ+06] Felfernig, Friedrich, Jannach & Zanker. An Integrated Environment for the Development of Knowledge-Based Recommender Applications, IJEC, 11(2), 2006, pp. 11-34.

[HJ09] Hegelich & Jannach. Effectiveness of different recommender algorithms in the mobile internet: A case study, ACM RecSys, 2009.

[MR09] Mahmood & Ricci. Improving recommender systems with adaptive conversational strategies. Hypertext 2009, pp. 73-82.

[McSherry05] McSherry. Retrieval Failure and Recovery in Recommender Systems, AIR 24(3-4), 2005, pp. 319-338.

[MRB04] Mirzadeh, Ricci & Bansal. Supporting User Query Relaxation in a Recommender System. EC-Web, 2004, pp. 31-40.

[PF04] Pu & Faltings. Decision Tradeoff using example critiquing, Constraints 9(4), 2004, pp. 289-310.