Machine Learning 520
Advanced Machine Learning

Lesson 8: Recommendation Systems

UNIVERSITY of WASHINGTON

Today's Agenda

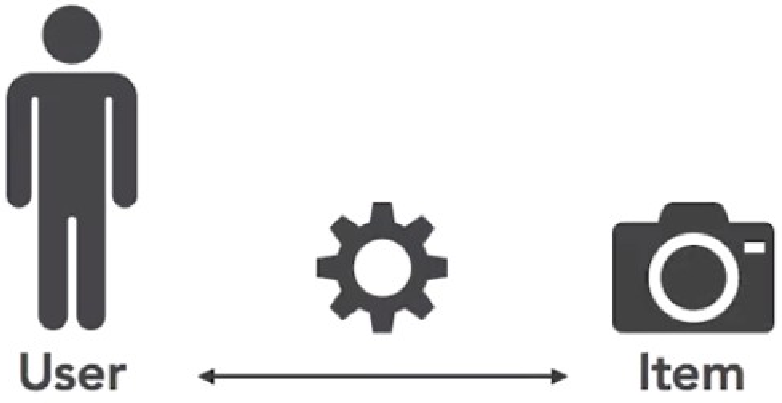
- What is a recommendation system?
- Content filtering
- Collaborative filtering
- Matrix factorization
- Hybrid system
- Evaluation the performance of recommendation systems

Learning Objectives

- Explain how content filtering works for recommendation.
- Explain the theory behind collaborative filtering.
- Use matrix factorization to estimate ratings (collaborative filtering).
- Explain how collaborative filtering and content filtering can be combined to make recommendations.
- Evaluate the performance of collaborative filtering algorithms.

Introduction

- Purpose: Find and recommend items that a user is most likely to be interested in



- Recommendation systems help people make decisions, be exposed to new content that they may be interested in but may not be aware of their existence
- Recommendation is based on implicit or explicit preferences of
  - Individuals
  - Groups
  - Population

Examples of Recommendation Engines

- Products recommendation: Amazon, ebay
- Movie recommendation: Netflix
- Music recommendation: Amazon music, Apple music
- Social connection recommendations: Facebook, LinkedIn, Instagram

Recommendation is not limited to products

Why do we need Recommendation Systems?

> **Information Overload:** There is too much information out there, one can easily be overwhelmed by choices. Recommendation systems help reduce the search space and cognitive overload

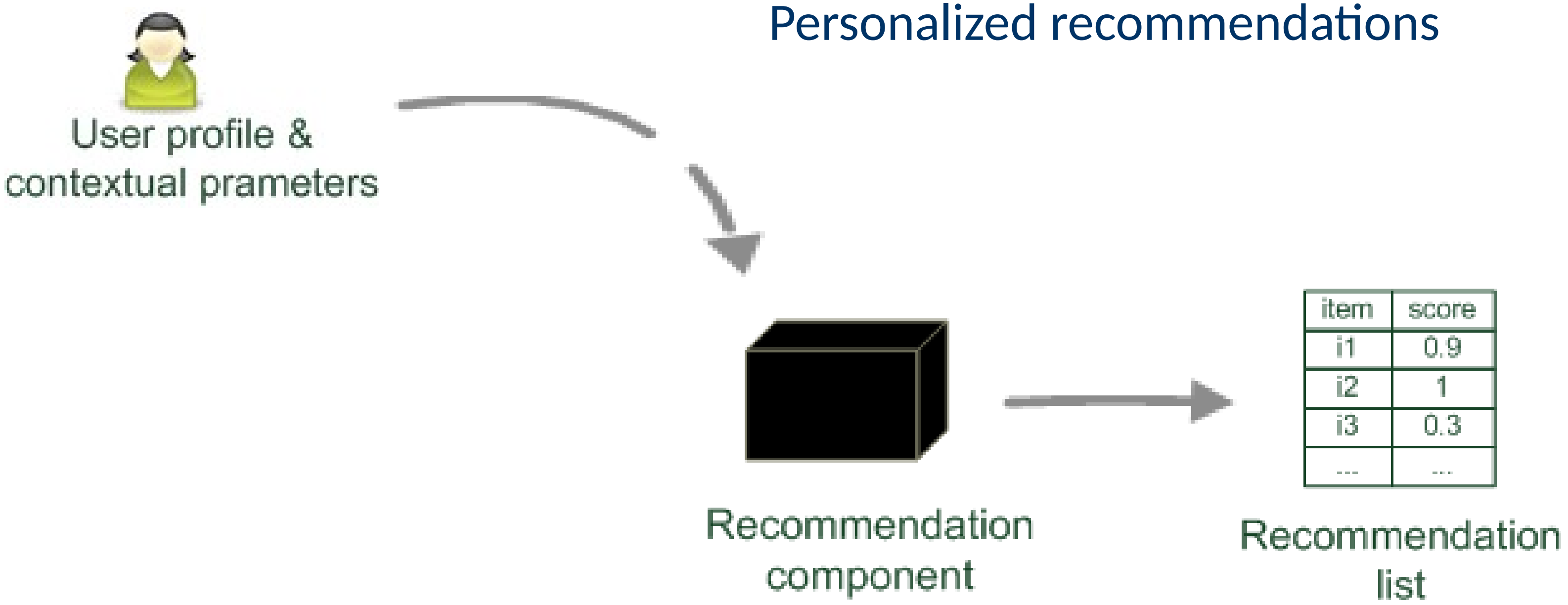
> **Item Discovery:** Provide guidance/assistance for item/concept discovery

> **Social Analogies**

- I like this book, you might also like this book since we have similar tastes
- Don’t go to watch that movie, you will hate it
- Since I liked this song, you will love these other songs

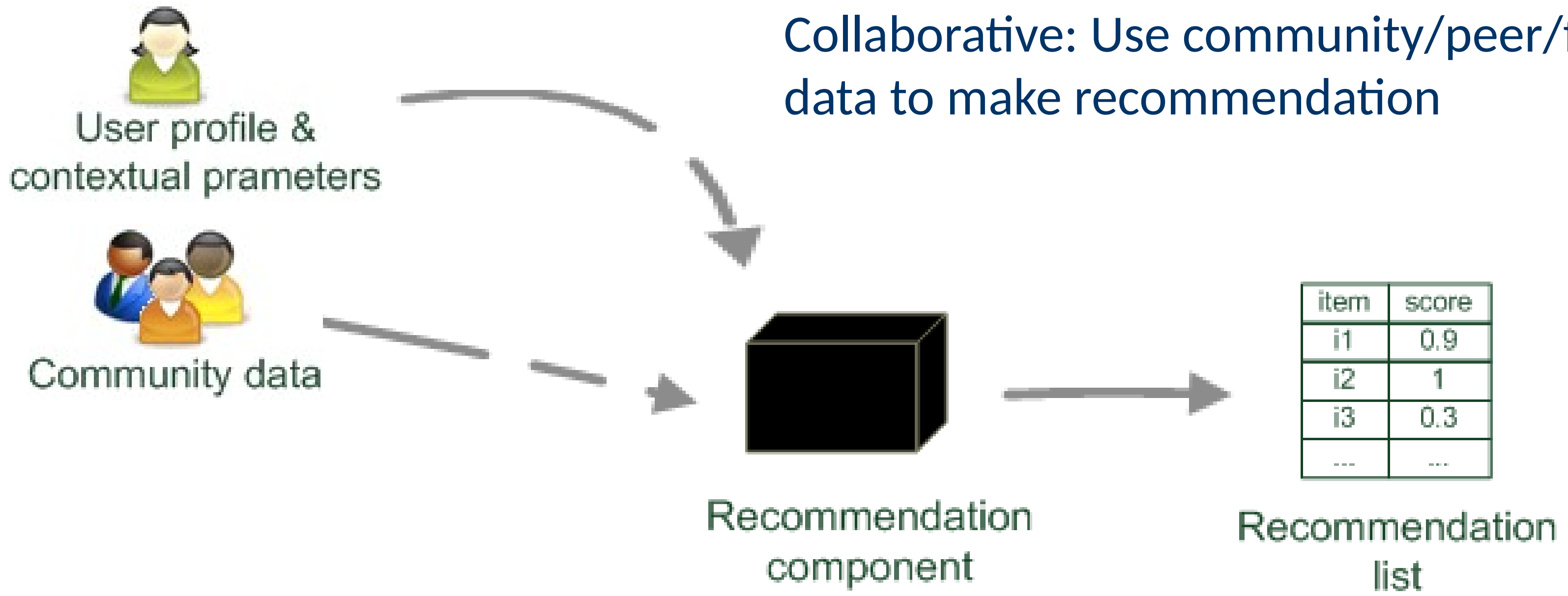
Paradigms of recommendation systems

Personalized recommendations



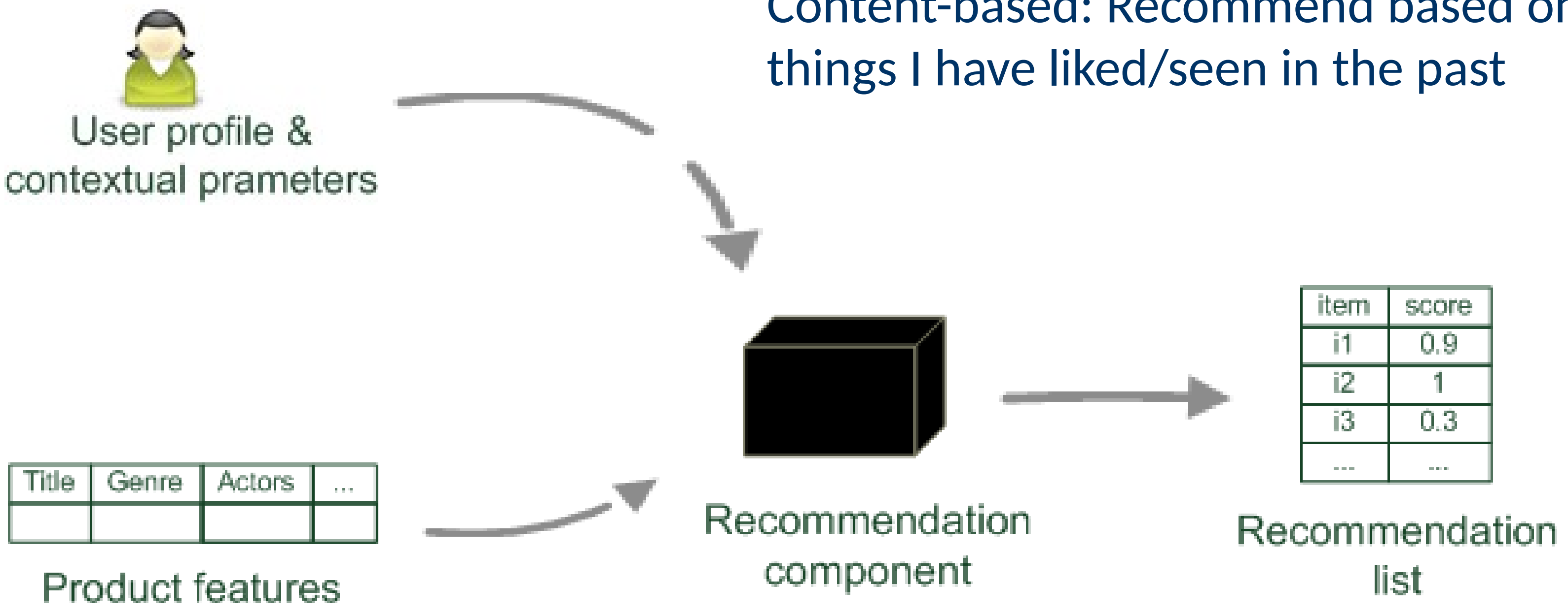
Paradigms of recommendation systems

Collaborative: Use community/peer/friends data to make recommendation



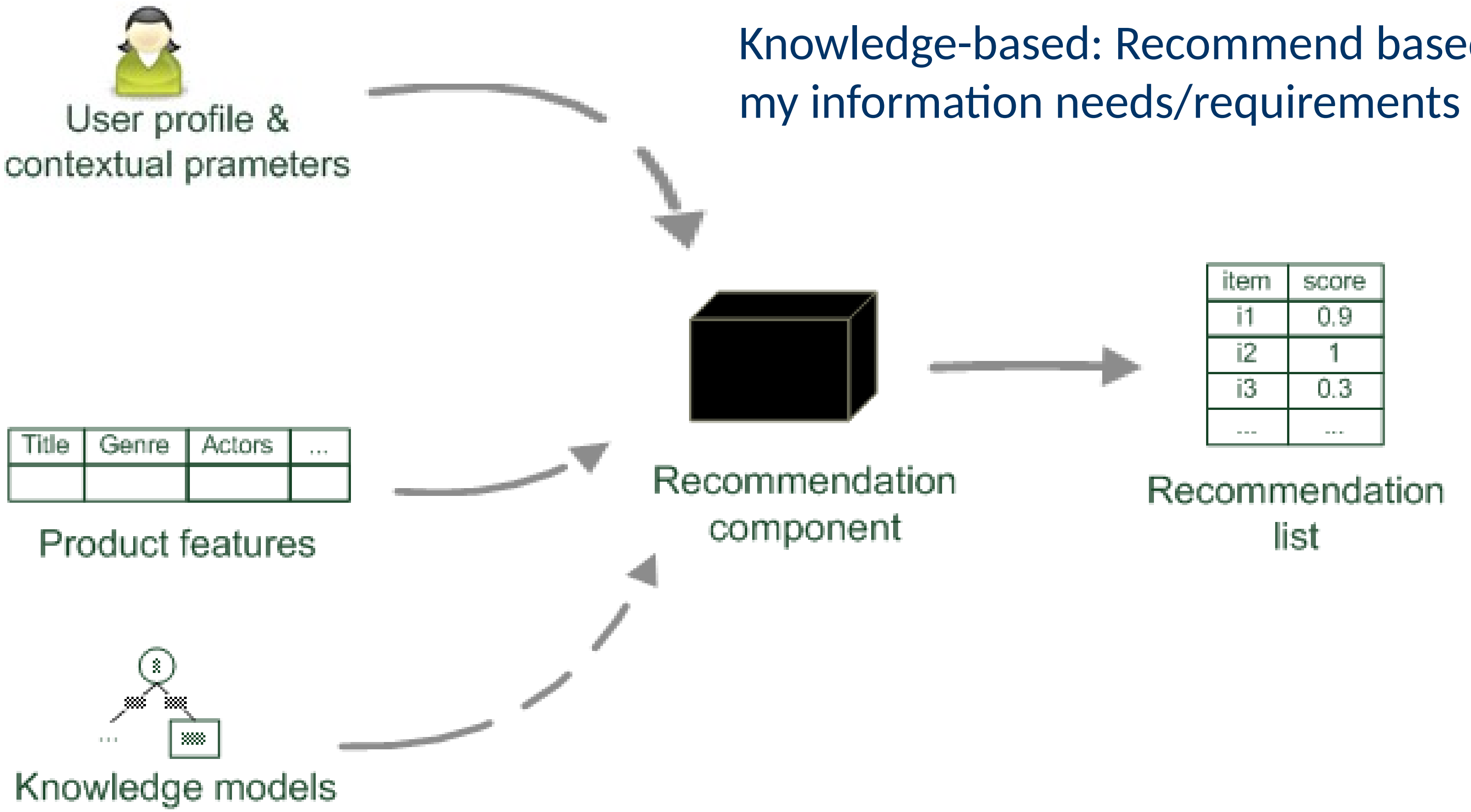
Paradigms of recommendation systems

Content-based: Recommend based on things I have liked/seen in the past



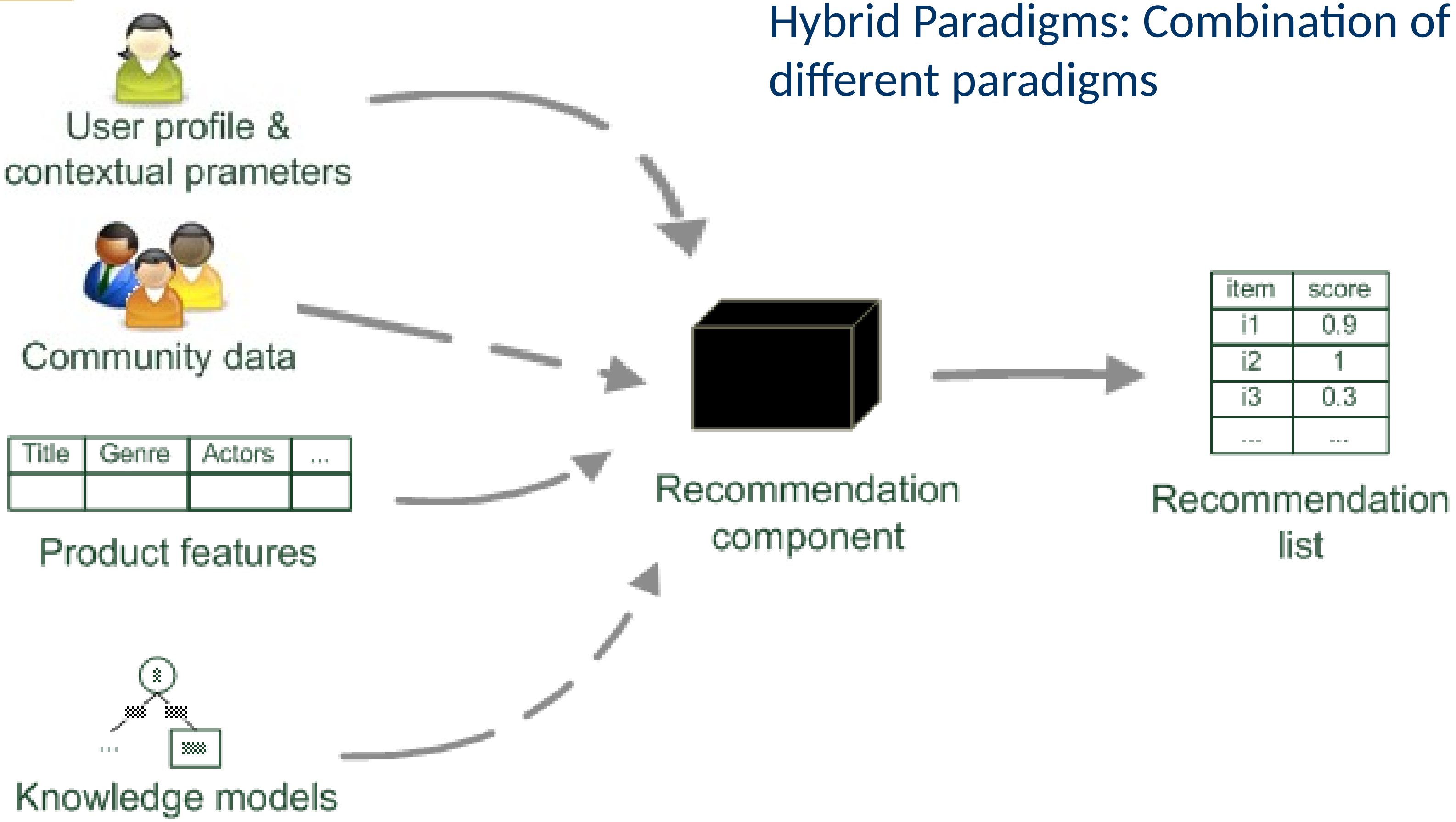
Paradigms of recommendation systems

Knowledge-based: Recommend based on my information needs/requirements



Paradigms of recommendation systems

Hybrid Paradigms: Combination of different paradigms



Main Types of Recommendation systems

	Pros	Cons
Collaborative	No hard-coded knowledge Serendipity of results Learns “natural” segments	Requires rating feedback cold start for new users and new items
Content-based	No community needed item-item comparison possible	Content descriptions necessary, cold start for new users, no surprises
Knowledge-based	Deterministic recommendations no cold-start	Knowledge engineering effort to bootstrap static short-term trends undetectable

Collaborative Recommendation Systems

Collaborative Filtering (CF)

>Main Idea

- Implicit or Explicit ratings information given by the user
- People who had similar interests in the past, will have similar interests in the future or for unknown things
- Use the "wisdom of the crowd”

>Application

- Widely used and deployed in industry
- Well-understood and thoroughly studied empirically
- Wide applicability in multiple domains

User-based NN collaborative filtering

- Nearest Neighbor Collaborative Filtering
  - Given a user (Alice) and an item ***i*** not yet seen by Alice
  - The *goal is to estimate Alice's rating for this item*:
    - find a set of users who liked the same items as Alice in the past **and** who have rated item ***i***
    - the average of their ratings to predict, if Alice will like item ***i***
  - Do this for all items Alice has not seen and recommend the best-rated

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

User-based NN collaborative filtering

>Foundational Questions

- How do we measure similarity?
- How many neighbors should we consider?
- How do we generate a prediction from the neighbors' ratings?

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

Measuring user similarity

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

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

a, b : users

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

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

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

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

sim = 0.85
sim = 0.70
sim = - 0.79

Recommendation Predictions

- Prediction function:

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

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

Improving Predictions

- 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", e.g., by linearly reducing the weight when the number of co-rated items is low
- Case Amplification
  - 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 vs. Model-based approaches

> Memory-based Approaches (nearest-neighbor or user-based collaborative filtering)
- The rating matrix is directly used to find neighbors / make predictions
- Does not scale for most real-world scenarios (Clever techniques to overcome these limitations)
- Large scale systems have tens of millions of customers and millions of items
- Collaborative Filtering is an example

> Model-based approaches (classification, clustering, and rule-based approach)
- 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

> Main idea:
  - Use the similarity between items (and not users) to make predictions

> Example:
  - Look for items that are similar to Item 5
  - Take Alice's ratings for these items to predict the rating for Item 5

	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

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

Scalability item-based filtering

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

Ratings in Item-Based Filtering

> Pure CF-based systems only rely on the rating matrix

> Explicit ratings

- Most commonly used (1 to 5, 1 to 7 Likert response scales)
- 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
- Can be used in addition to explicit ones; question of correctness of interpretation

Data sparsity problems

- Cold start problem
  - How to recommend new items? What to recommend to new users?
- Straightforward approaches
  - Ask/force users to rate a set of items
  - Data Driven Methods e.g., content-based, demographic or simply non-personalized
- 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 user ***u*** who 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

More model-based approaches

- > Matrix factorization techniques, statistics
  - + singular value decomposition, principal component analysis
- > Association rule mining
  - + Use Association rules to find common co-occurring items for recommendation
- > Probabilistic models
  - + Clustering models, Bayesian networks, probabilistic Latent Semantic Analysis

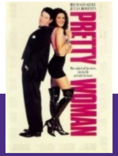
Vector Space Models

- 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, as well as factors not easily described in human language
  - 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, etc.

Matrix factorization

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

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

					
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	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

Power of Matrix Factorization

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

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

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

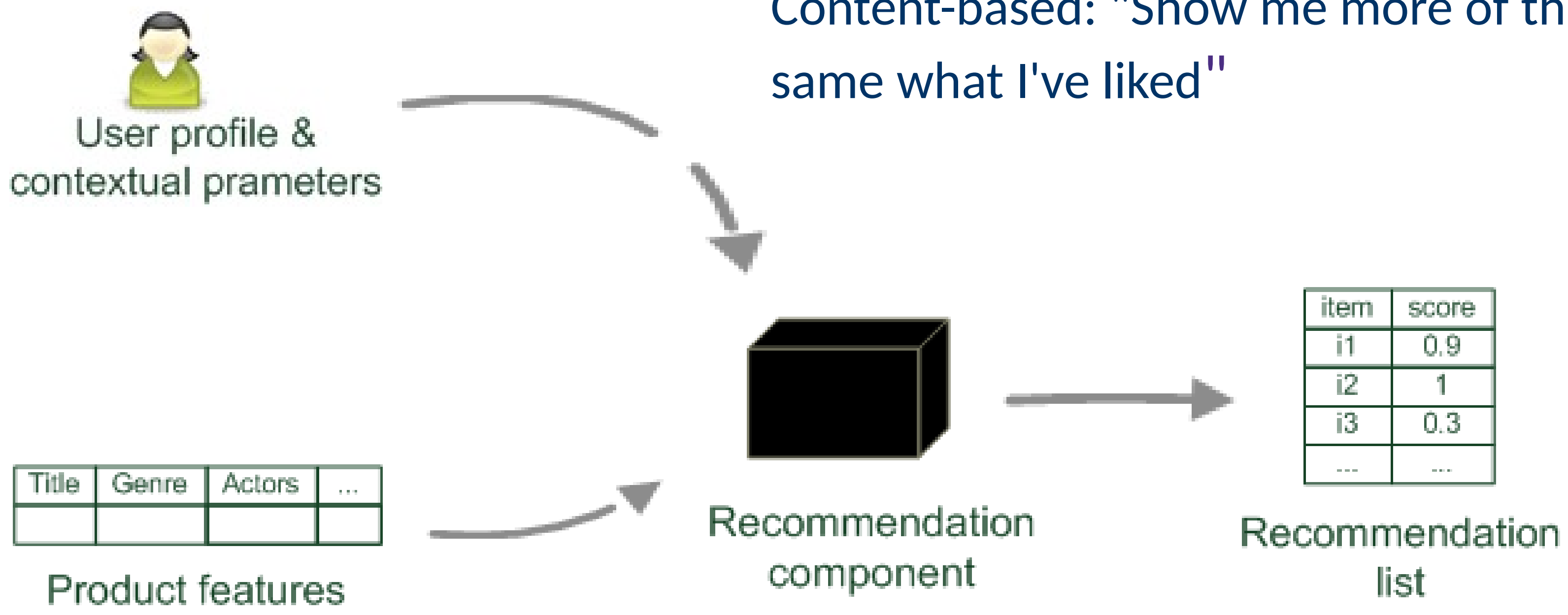
Content-Based Recommendations

Content-based recommendation

- Collaborative filtering does NOT require any information about the items:
  - However, it might be reasonable to exploit such information
  - E.g. recommend fantasy novels to people who liked fantasy novels in the past
- What do we need:
  - Some information about the available items such as the genre ("content")
  - Some sort of *user profile* describing what the user likes (the preferences)
- The task:
  - Learn user preferences
  - Locate/recommend items that are "similar" to the user preferences

Paradigms of recommendation systems

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



What is *Content*?

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

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

Content representation and item similarities

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

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

- Compute the similarity of an unseen item with the user profile based on the keyword overlap (e.g. using the Dice coefficient)
- $sim(b_i, b_j) =$

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

- Simple keyword representation has its problems
  - In particular when automatically extracted because
    - Not every word has similar importance
    - Longer documents have a higher chance to have an overlap with the user profile
- Standard measure: TF-IDF
  - Encodes text documents as weighted term vector
  - TF: Measures, how often a term appears (density in a document)
    - Assuming that important terms appear more often
    - Normalization has to be done in order to take document length into account
  - IDF: Aims to reduce the weight of terms that appear in all documents

Improvements

- Side note: Conditional independence of events does in fact not hold
  - “New”/ “York“ and “Hong” / “Kong"
  - Still, good accuracy can be achieved
- Boolean representation simplistic
  - 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
- Classification algorithms can also be used

Limitations of content-based recommendation methods

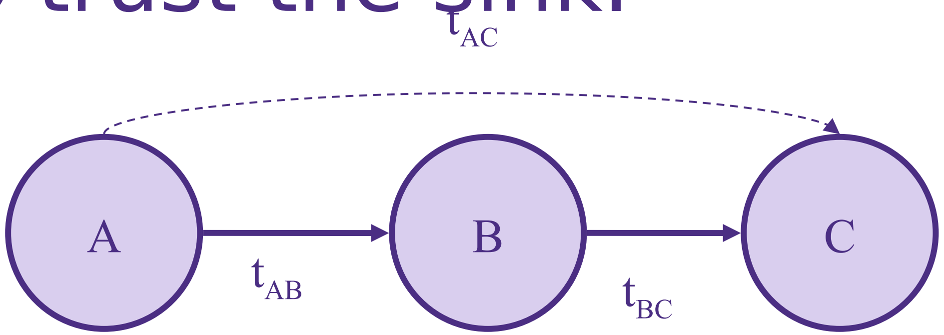
- Keywords alone may not be sufficient to judge quality/relevance of a document or web page
  - Up-to-dateness, usability, aesthetics, writing style
  - Content may also be limited / too short
  - Content may not be automatically extractable (multimedia)
- Ramp-up phase required
  - Some training data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

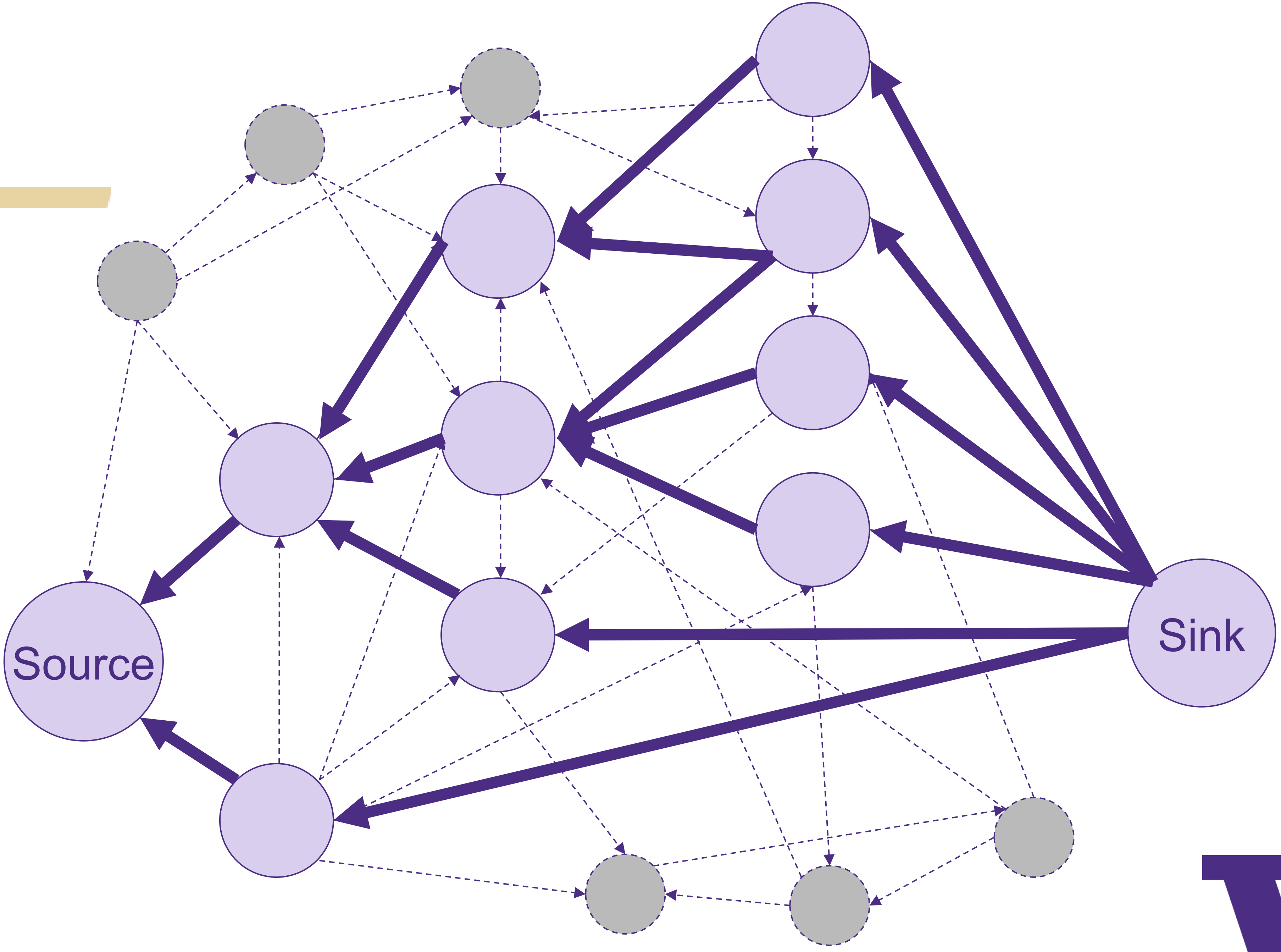
Network-Based Recommendations

Recommendation in Social Networks

Inferring Trust

The Goal: Select two individuals - the *source* (node A) and *sink* (node C) - and recommend to the source how much to trust the sink.




Sink
Source

TidalTrust Algorithm

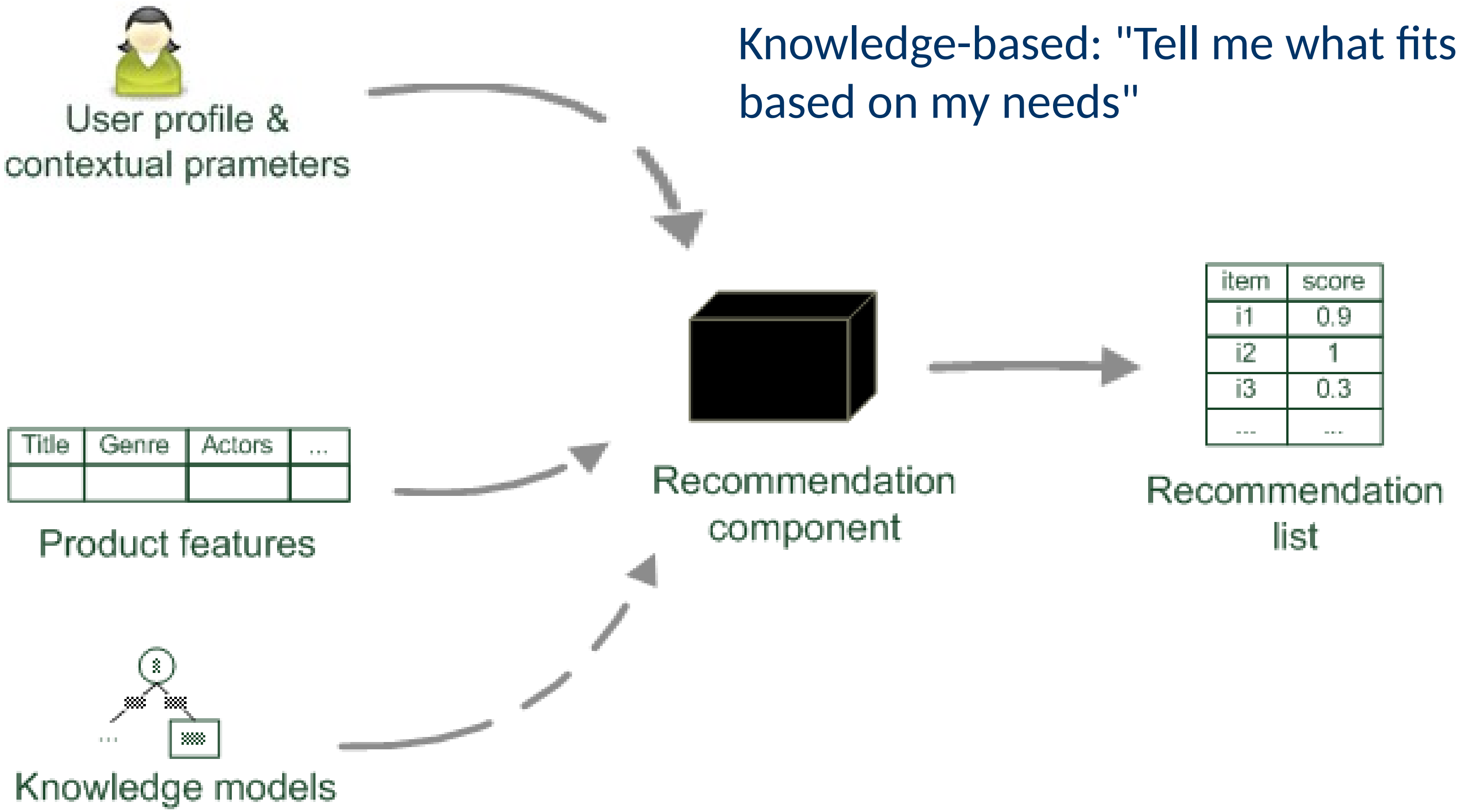
$$t_{is} = \frac{\displaystyle\sum_{j \in adj(j) \mid t_{ij} \geq max} t_{ij} t_{js}}{\displaystyle\sum_{j \in adj(j) \mid t_{ij} \geq max} t_{ij}}$$

Knowledge-Based Recommendation Systems

Why do we need knowledge-based recommendation?

Knowledge-based recommendation

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



Knowledge-based recommendation

Constraint-based recommendation I

- A knowledge-based RS formulated as constraint satisfaction problem

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

where

- X_I, X_U: Variables describing items and user model with domain D (e.g. lower focal length, purpose)
- KB: Knowledge base comprising constraints and domain restrictions (e.g. **IF** purpose="*on travel*" **THEN** lower focal length < *28mm*)
- SRS: Specific requirements of a user (e.g. purpose = "*on travel*")
- I: Product catalog (e.g. (id=1 ˆ lfl = *28mm*) ˇ (id=2 ˆ lfl= *35mm*) ˇ …)

- Solution: Assignment tuple θ assigning values to all variables X_I is satisfiable

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

Item ranking

> Multi-Attribute Utility Theory (MAUT)
  - Each item is evaluated according to a predefined set of dimensions that provide an aggregated view on the basic item properties

> E.g. <u>quality</u> and <u>economy</u> are dimensions in the domain of digital cameras

id	value	quality	economy
price	≤250 >250	5 10	10 5
mpix	≤8 >8	4 10	10 6
opt-zoom	≤9 >9	6 10	9 6

Customer-specific item utilities with MAUT

Customer interests:

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

*

Item utilities:

quality	economy	utility: cu_1	utility: cu_2
P1 Σ(5,4,6,6,3,7,10) = 41	Σ (10,10,9,10,10,10,6) = 65	45.8 [8]	55.4 [6]
P2 Σ(5,4,6,6,10,10,8) = 49	Σ (10,10,9,10,7,8,10) = 64	52.0 [7]	58.0 [1]
P3 Σ(5,4,10,6,10,10,8) =	Σ (10,10,6,10,7,8,10) =	54.6 [5]	57.8 [2]

**

$$utility(p) = \sum_{j=1}^{\#(dimensions)} \underbrace{interest(j)}_{*} * \underbrace{contribution(p,j)}_{**}$$

Constraint-based recommendation II

- **BUT**: What if no solution exists?
  - $KB \cup I$ not satisfiable ☾ debugging of knowledge base
  - $SRS \cup KB \cup I$ not satisfiable but $KB \cup I$ correct ☾ 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: $CS \subseteq SRS : CS \cup KB \cup I$ not satisfiable

Ask user

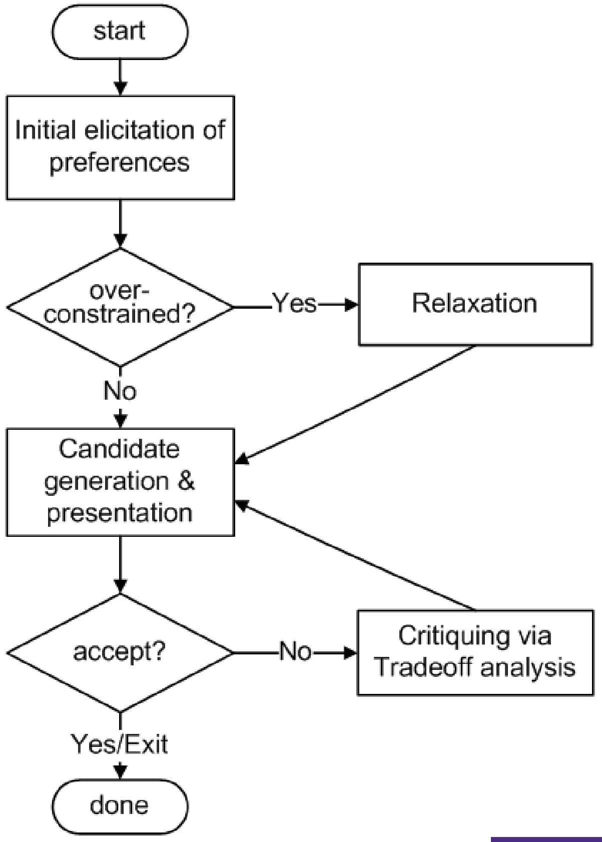
- Computation of minimal revisions of requirements
  - Do you want to relax your brand preference?
    - Accept *Panasonic* instead of *Canon* brand
  - What are other requirements?
    - Maximize features or cost?

Constraint-based recommendation III

- More variants of recommendation task
  - Customers maybe not know what they are seeking
  - Find "diverse" sets of items
    - Notion of similarity/dissimilarity
    - Idea that users navigate a product space
    - We do not want to recommend too many things for the user to handle; there might be cognitive limitations.
  - 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

Conversational strategies

- Process consisting of multiple conversational moves
  - Resembles natural sales interactions
  - Not all user requirements known beforehand
  - Customers are rarely satisfied with the initial recommendations
- Different styles of preference elicitation:
  - Free text query interface
  - Asking technical/generic properties
  - Images / inspiration
  - Proposing and Critiquing



Limitations of KB Recommendations

>Cost of knowledge acquisition

- From domain experts
- From users
- Remedy: exploit web resources

>Accuracy of preference models

- Very fine granular preference models require many interaction cycles with the user or sufficient detailed data about the user
- Remedy: use collaborative filtering, estimates the preference of a user

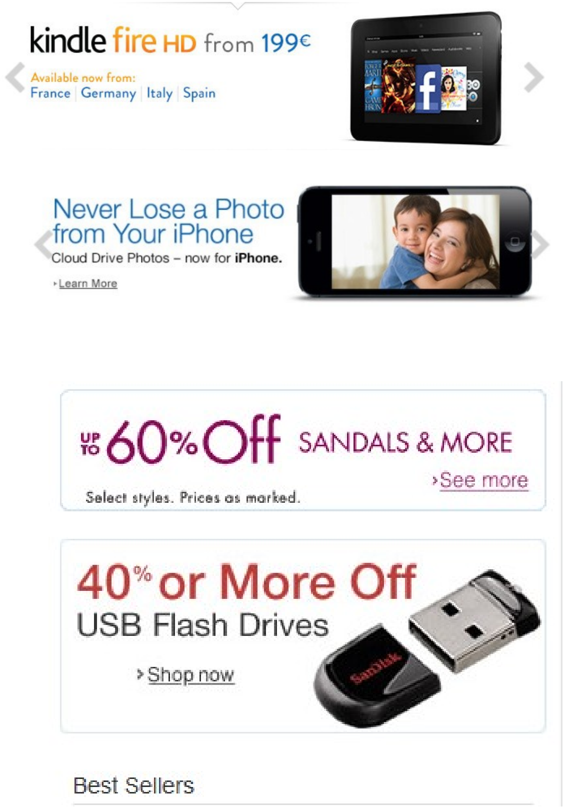
However: preference models may be instable

+ E.g. asymmetric dominance effects and decoy items

Evaluating Recommendation Systems

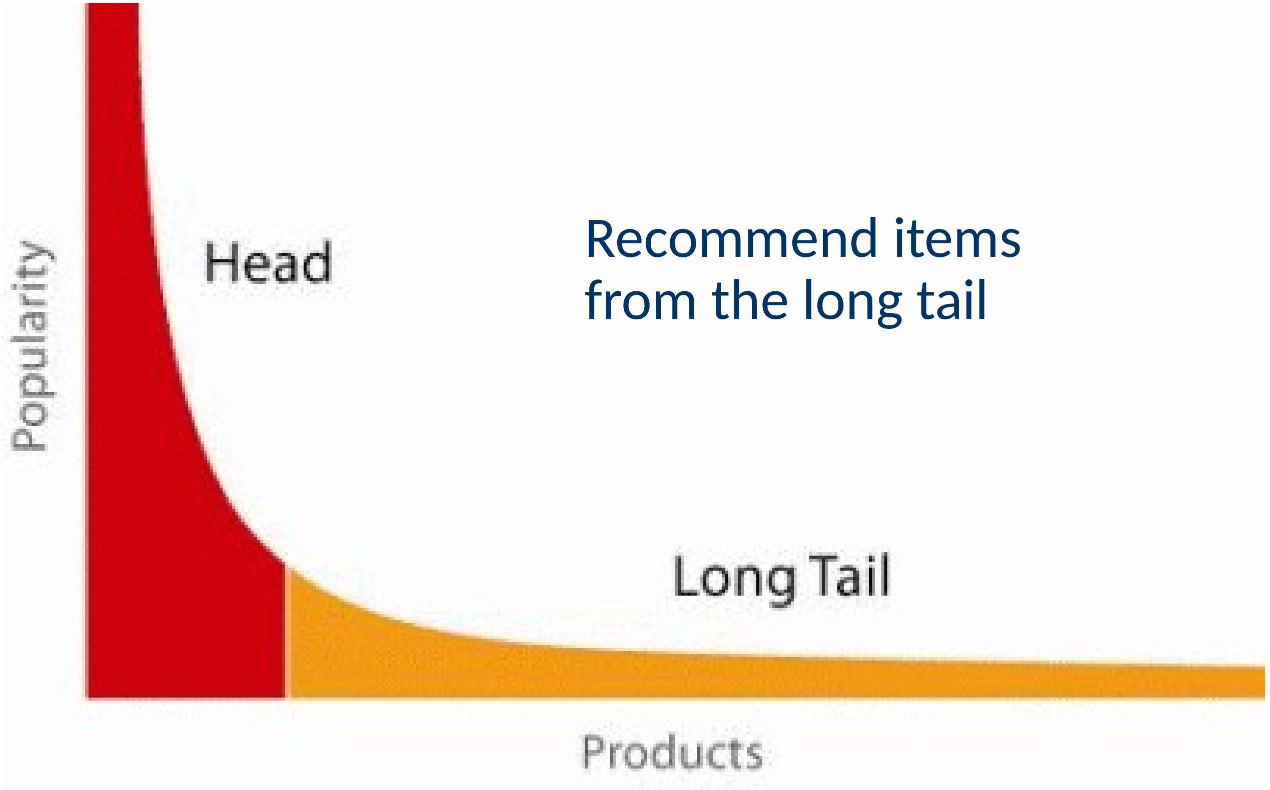
What is a good recommendation?

- The definition of what constitutes a good recommendation is context dependent
- Total sales numbers
- Promotion of certain items
- Click-through-rates
- Interactivity on platform
- Customer return rates
- Customer satisfaction and loyalty



Purpose and success criteria

When does a RS do its job well?



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

Purpose and success criteria (2)

- Prediction perspective
  - Predict to what degree users like an item
  - Most popular evaluation scenario in research

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

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

Metrics: Rank Score – position matters

For a user:

Actually good
Item 237
Item 899

hit

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

> **Rank Score** extends recall and precision to take the positions of correct items in a ranked list into account

- Particularly important in recommendation systems as lower ranked items may be overlooked by users
- Learning-to-rank: Optimize models for such measures (e.g., AUC)

Accuracy measures

> Datasets with items rated by users
  - MovieLens datasets 100K-10M ratings
  - Netflix 100M ratings

> Historic user ratings constitute ground truth

> Metrics measure error rate
  - 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

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

Offline experimentation example

- Netflix competition
  - Web-based movie rental
  - Prize of $1,000,000 for accuracy improvement (RMSE) of 10% compared to Netflix's 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
- Today
  - Rating prediction only seen as an additional input into the recommendation process

An imperfect world

- Offline evaluation is the cheapest variant
  - Still, gives us valuable insights
  - Allows one to compare results
- Dangers and trends:
  - Domination of accuracy measures
  - Focus on small set of domains (40% on movies in CS)
- Alternative measures:
  - Diversity, Coverage, Novelty, Familiarity, Serendipity, Popularity, Concentration effects (Long tail)

Psychological Factors

Phenomenon/Effect	Description
Decoy effects	Additional irrelevant (inferior) items in an item set significantly influence the selection behavior
Primacy/recency effects	Items at the beginning and the end of a list are analyzed significantly more often/deeply than items in the middle of a list
Framing effects	The way in which different decision alternatives are presented influences the final decision taken
Priming	If specific decision properties are made more available in memory, this influences a consumer's item evaluations (background priming)
Defaults	Preset options bias the decision process

Decoy: asymmetric dominance effect

Product	A	B	D
price per month	30	20	50
download limit	10GB	6GB	9GB

- Product *A* dominates *D* in both dimensions (price and download limit)
- Product *B* dominates alternative *D* in only one dimension (price)
- The additional inclusion of *D* into the choice set could trigger an increase of the selection probability of *A*

Personality

- Different personality properties pose specific requirements on the design of recommender user interfaces

- Some personality traits are more susceptible to heuristic simplifications

- Provide various interfaces

Personality traits

Theory	Description
Internal vs. external Locus of control (LOC)	Externally influenced users need more guidance; internally controlled users want to actively and selectively search for additional information
Need for closure	Describes the individual pursuit of making a decision as soon as possible
Maximizer vs. satisficer	Maximizers try to find an optimal solution; satisficers search for solutions that fulfill their basic requirements

Dark Side of Recommendation Systems

- Recommendation is not just about products but about *things of interest* in general
- If the objective of the recommendation is to maximize clicks then other factors like social impacts of the click will be ignored (e.g., recommendation of fake news articles based on your likes or history)
- Thus recommendations can create an echo chamber

Conclusion

- A wide variety of techniques are used in recommendation systems
- Recommendation systems reduce cognitive overload, make object discovery easier, etc.
- Three major paradigms of recommendation systems:
  - Collaborative
  - Content-Based
  - Knowledge-Based
- The hybrid approach is a combination of the three approaches