Analysis of massive data sets

http://www.fer.hr/predmet/avsp

Prof. dr. sc. Siniša Srbljić

Doc. dr. sc. Dejan Škvorc

Doc. dr. sc. Ante Đerek

Faculty of Electrical Engineering and Computing
Consumer Computing Laboratory

Recommender Systems

Marin Šilić, PhD

Outline

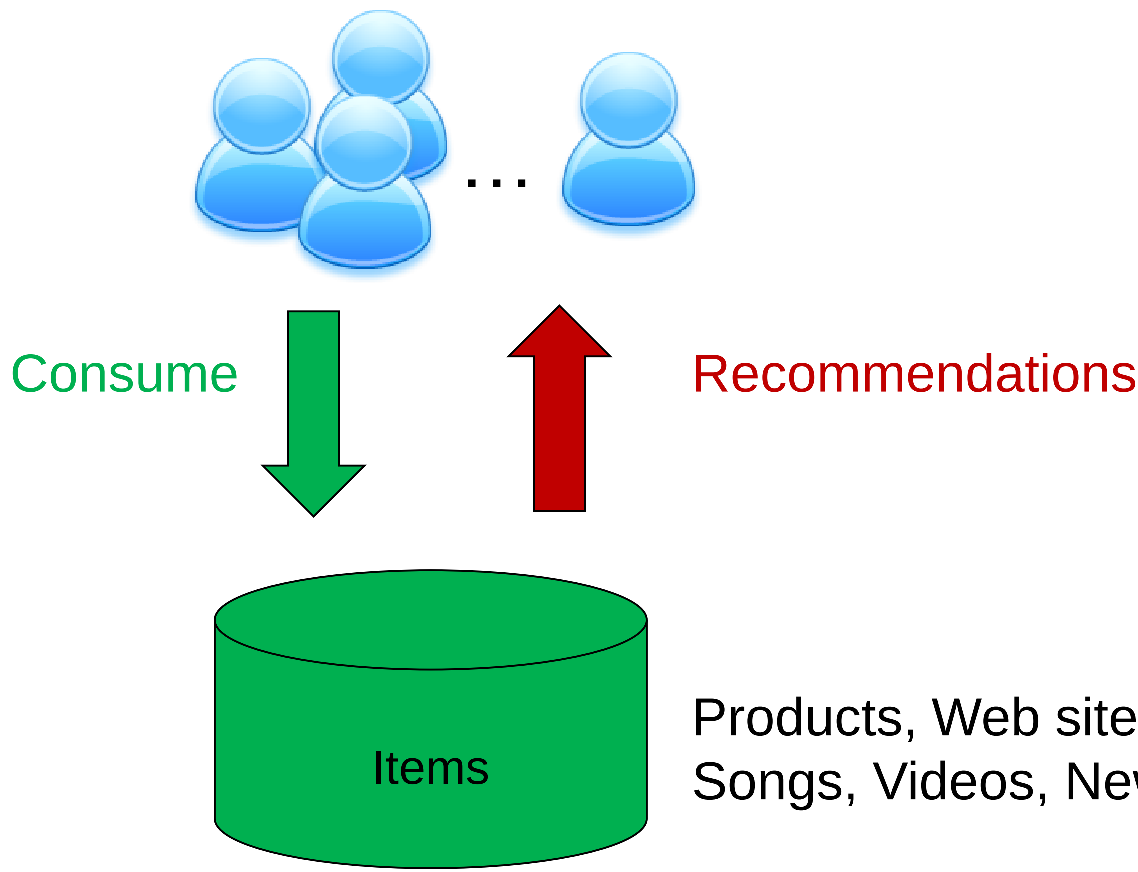
- Motivation
- Formal Definition
- Content-based Recommendation
- Collaborative Filtering
- Remarks, Practice & Advices

Motivation

- **Example**
  - Music CDs web shop
  - User A
    - Buys Metallica CD
    - Buys Iron Maiden CD
  - User B
    - Does search on Metallica
    - Recommender suggests Iron Maiden from the data collected about User A

Motivation

- Recommender Systems



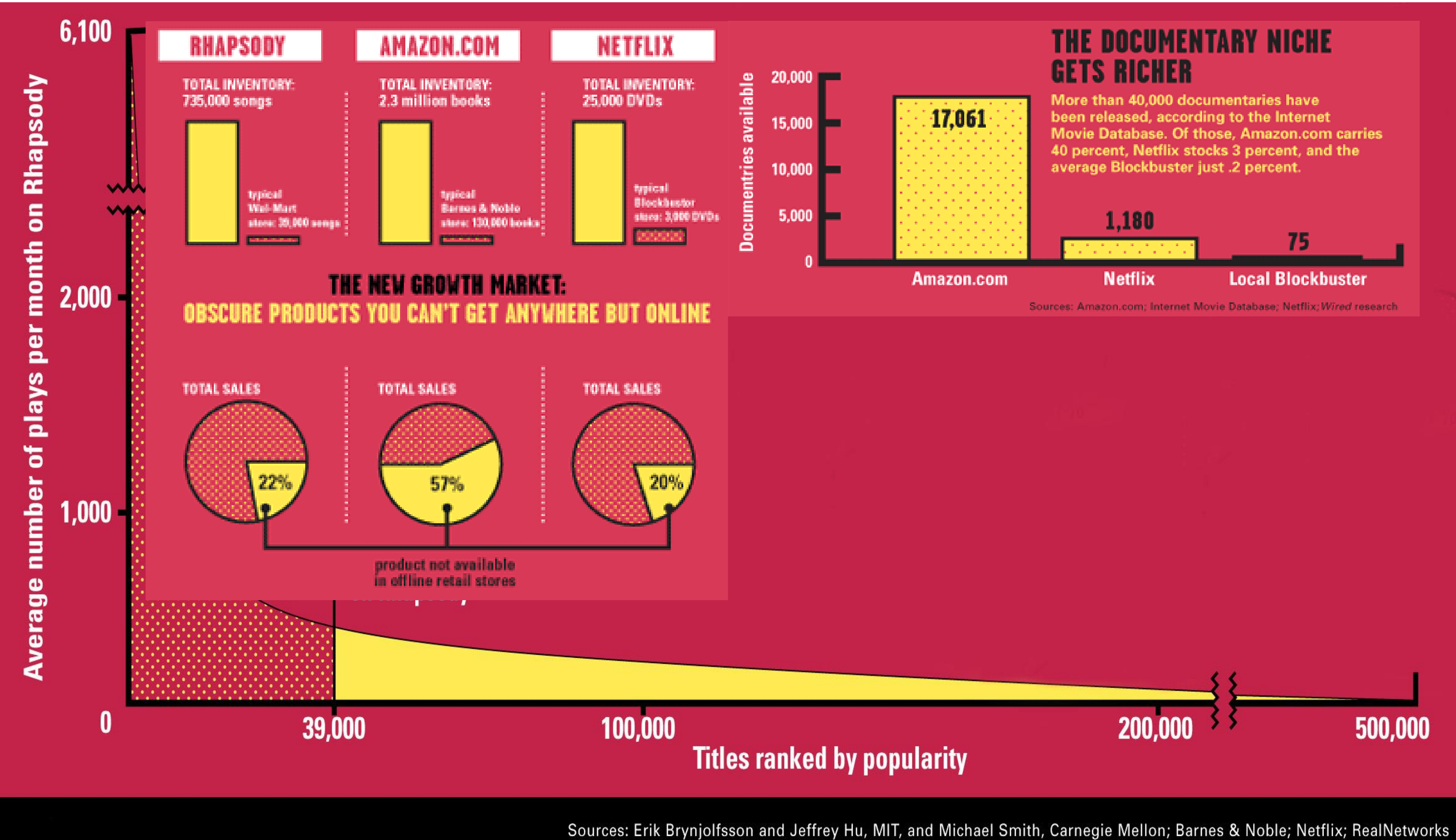
Products, Web sites,
Songs, Videos, News…

Examples

Motivation

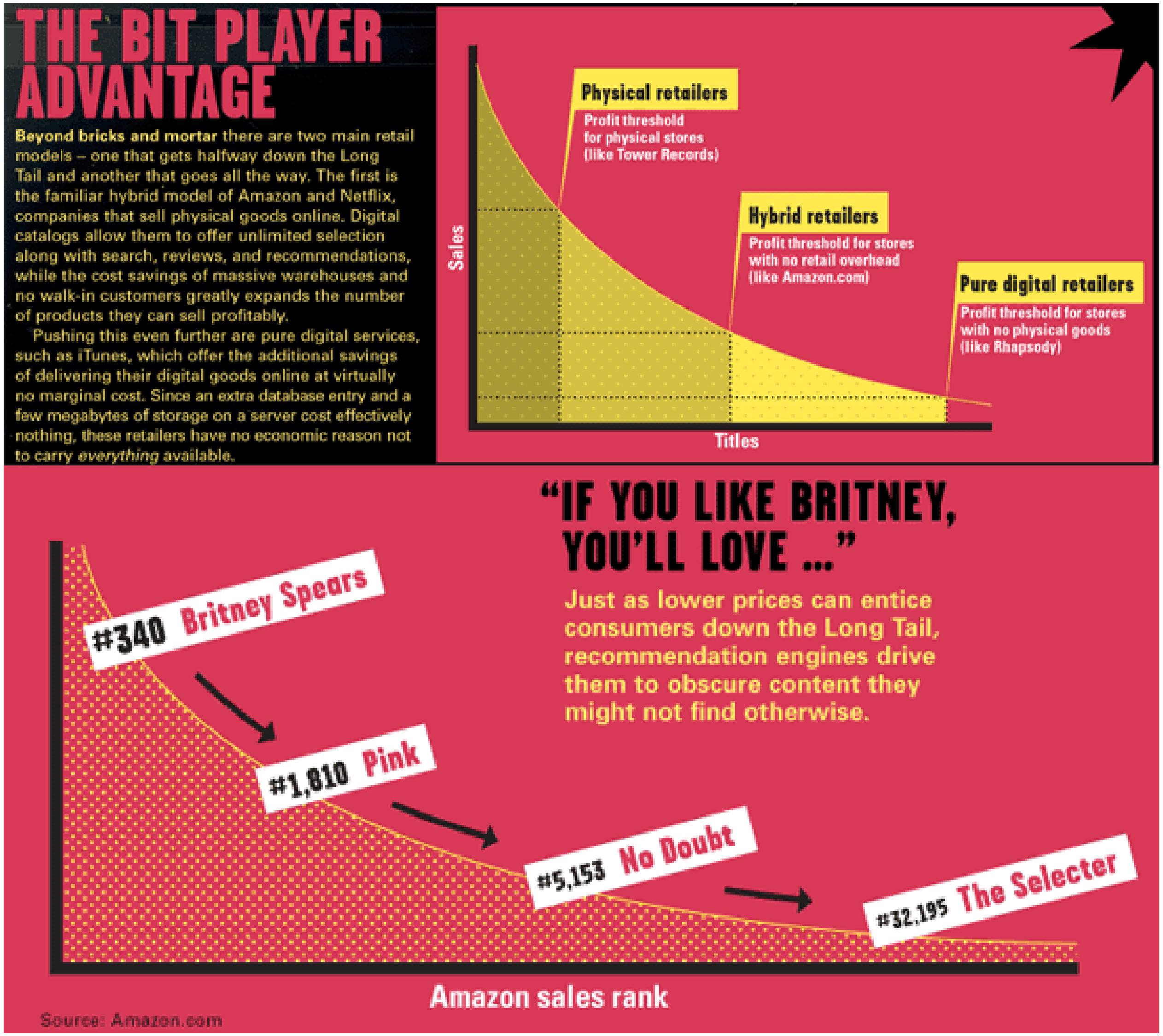
- **From Scarcity to Abundance**
  - Shelf space is a limited resource to traditional retailers
    - Similar: TV advertisement, movie theaters etc.
  - Web enables almost zero-cost dissemination of information about products
    - No issue with limited space
    - Moreover, product information can be personalized
  - The more choice there is, better filter are required
    - Recommendation engines
    - How ***Into Thin Air*** made ***Touching the Void*** a bestseller http://archive.wired.com/wired/archive/12.10/tail.html

Motivation



Source: Chris Anderson (2004)

Motivation



Read & Learn More

Motivation

- **Types of Recommendations**
  - Editorial and hand selected
    - List of favorites
    - List of “must see” items
  - Simple aggregates
    - Top 10
    - Most Popular
    - Most Recent
  - Personalized recommendations
    - Amazon, EBay, YouTube etc…

Formal Definition

- **Formal model**
  - X is a set of customers
  - Y is a set of items
  - R is a set of ratings

  - We define a utility function $u\colon X \times Y \rightarrow R$
  - R is a totally ordered set
    - E.g. 0-5 stars, real number $\in [0,1]$ etc.

Formal Definition

- **Utility Matrix**
  - Then, we define a utility matrix which maps users and ratings

	Avatar	LOTR	Matrix	Pirates
Alice				
Bob				
Carol				
David				

Formal Definition

- **Utility Matrix**
  - Then, we define a utility matrix which maps users and ratings

	Avatar	LOTR	Matrix	Pirates
Alice	1.0		0.2	
Bob		0.5		0.3
Carol	0.2		1.0	
David				0.4

Formal Definition

- **Main Challenges**
  - Challenge #1: Collecting “known” ratings in matrix
    - How to collect the initial data?
  - Challenge #2: Utilize known ratings in order to estimate/predict the unknown ratings
    - Focus on high unknown ratings
    - The idea is to rather extract the information what a particular user likes than dislikes
  - Challenge #3: Evaluate prediction methods
    - How to measure accuracy & performance of recommendation methods

Formal Definition

- **Challenge #1: Collecting Initial Ratings**
  - Explicit
    - Ask people to rate items
    - Not effective in practice
    - Users don’t like to provide feedback explicitly
  - Implicit
    - Learn ratings from users’ actions
      - E.g. purchase implies high rating
    - How can one implicitly detect low ratings?

Formal Definition

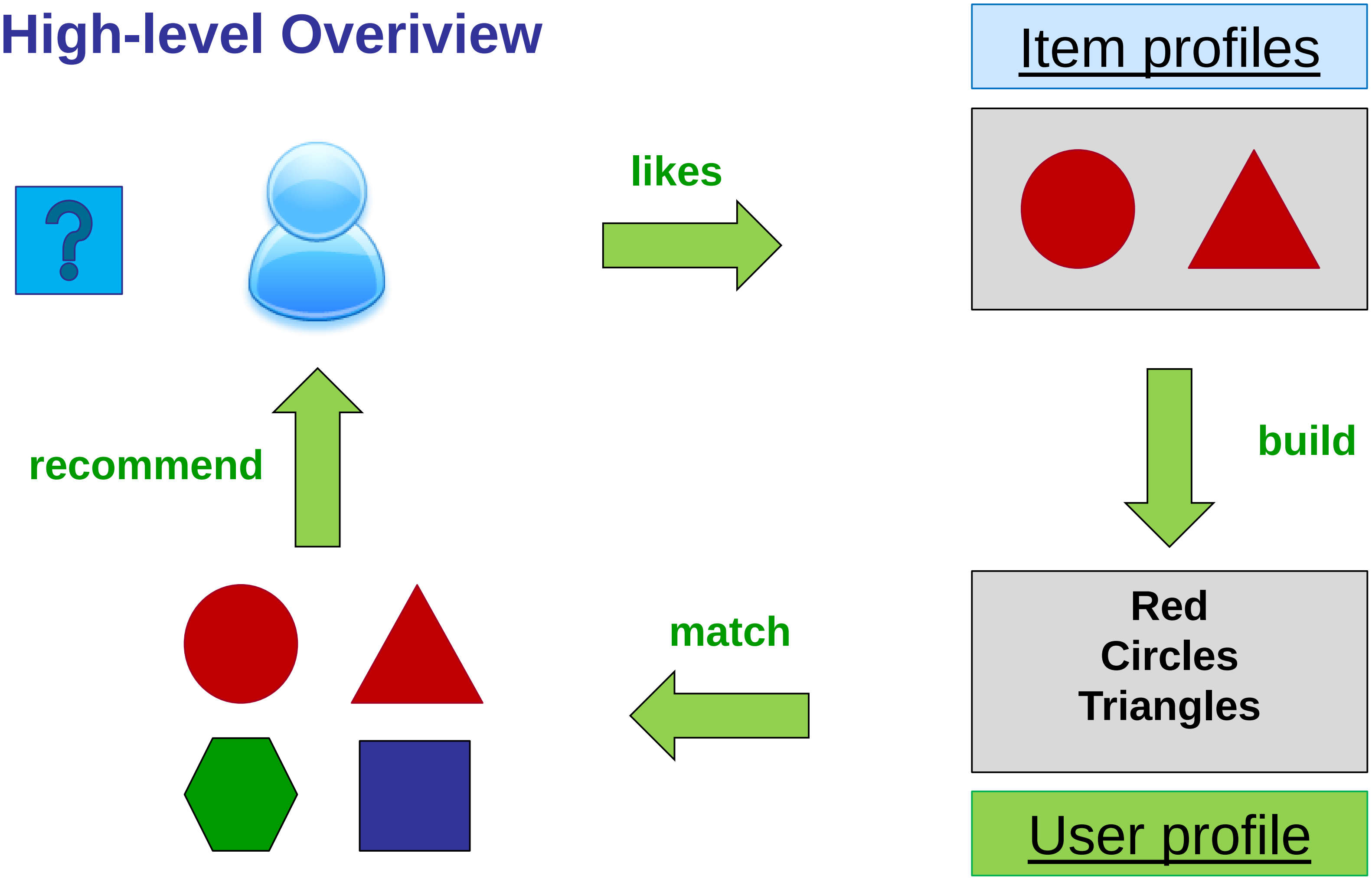
- **Challenge #2: Utilize available ratings to predict missing ones**
  - Data sparsity: Utility matrix U is extremely sparse
    - Ordinary user rates only a very limited subset of items
  - Cold start issue:
    - New users don't have rating history
    - New items don't have any ratings
  - Most effective approaches to overcome the challenges in recommender systems
    - Content based recommendation
    - Collaborative filtering
    - Latent factor analysis

Content-based Recommender Systems

- **Main idea**
  - Recommend user A items that are similar to previous items highly rated by user A
- **Example**
  - Movie recommendations
    - Recommends movies with same actor(s) director, genre, etc.
  - Websites, blogs, news
    - Recommend other sites with similar content

Content-based Recommender Systems

- **High-level Overiview**



Content-based Recommender Systems

- **Item Profiles**
  - For each item, create an item profile
  - Profile is a set (vector) of features
    - Movies: author, title, director, genre…
    - Text: Set of “important” words in document
  - How to pick important features?
    - Usual heuristic from text mining is TF-IDF
      (Term frequency * Inverse Document Frequency)
    - Term … Feature
    - Document … Item

Content-based Recommender Systems

- **TF-IDF**
  - f_{ij} is a frequency of term (feature) i in a doc (item) j

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

  - n_i is a number of docs that contain item i
  - N is a total number of docs

$$IDF_i = \log \frac{N}{n_i}$$

  - TF-IDF score $w_{ij} = TF_{ij} \times IDF_i$
  - Document profile:
    - set of words with highest TF-IDF scores along with their scores

Content-based Recommender Systems

- **User Profiles and Prediction**
  - User profile possibilities
    - Weighted average of rated item profiles
    - Variation: weight by difference from average rating for item
    - …
  - Prediction heuristic
    - Estimate $u(i,j)$ for user i and item j:

$$u(i,j) = cosine(i,j) = \frac{i \cdot j}{||i|| \cdot ||j||}$$

Content-based Recommender Systems

- **User Profiles Example**
  - User has rated 5 movies
    - 2 starring actor A, ratings 3 and 5
    - 3 starring actor B, ratings 1, 2 and 4
  - Let's build user profile
    - First, notice that ratings are relative,
      - For one user rating 4 might be a high rating
      - For some other user rating 4 might be an average rating
    - So, first normalize ratings by subtracting the mean which is 3
      - Actor A, 0 and 2, Actor B, -2, -1 and 1
    - Weight of feature A becomes (2 + 0) / 2 = 1
    - Weight of feature B becomes (-2 -1 + 1) / 3 = -2 / 3
    - User profile is A + (-2/3) * B

Content-based Recommender Systems

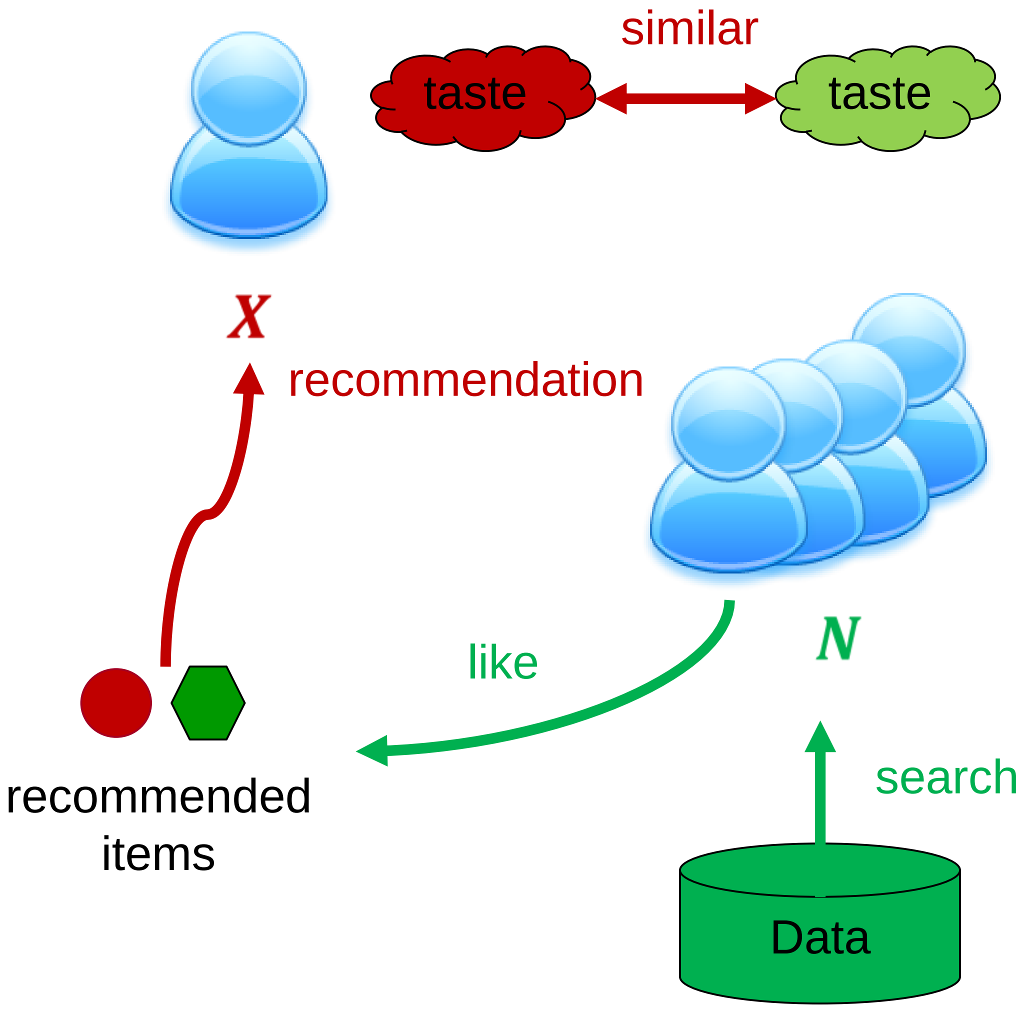
- **Pros: Content-based RS**
  - No need for data on other users
    - No cold-start and sparsity issues
  - It can recommend to users with unique tastes
  - It can recommend new & unpopular items
    - No first-rater issue
  - Transparency
    - It can provide transparent explanation about content features that caused an item to be recommended

Content-based Recommender Systems

- **Cons: Content-based RS**
  - Finding appropriate features is difficult and requires domain-specific knowledge
    - E.g., movies, images, books etc.
  - It can not recommend for new users
    - How to build user profile for a new user?
  - Overspecialization
    - Recommends only items that match user's content profile
    - People might have multiple interests
    - **It can not utilize ratings from other users**

Collaborative Filtering

- Imagine a user X
- Find a set N containing users whose ratings are similar to X's ratings
- Estimate X's ratings based on ratings of users in N



Collaborative Filtering

- **Finding Similar Users**
  - Let vector r_x represent user X's ratings
  - Jaccard similarity measure
    - Issue: It does not consider the values of ratings
  - Cosine similarity measure
    - $sim\_cosine(x, y) = cosine(r_x, r_y) = \frac{r_x \cdot r_y}{\|r_x\| \cdot \|r_y\|}$
    - Issue: It threats missing ratings as "negative"
  - Pearson correlation coefficient
    - Let S_{xy} be the set of items rated by both user X and Y

$$sim\_pearson(x, y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2 \sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

Collaborative Filtering

Similarity Metric

	I1	I2	I3	I4	I5	I6	I7
A	4			5	1		
B	5	5	4				
C				2	4	5	

- Intuitively we want: $sim(A, B) > sim(A, C)$
  - Jaccard similarity produces: 1/5 < 2/4

Collaborative Filtering

Similarity Metric

	I1	I2	I3	I4	I5	I6	I7
A	4	0	0	5	1	0	0
B	5	5	4	0	0	0	0
C	0	0	0	2	4	5	0

- Intuitively we want: $sim(A, B) > sim(A, C)$
  - Jaccard similarity produces: 1/5 < 2/4
  - Cosine similarity produces: 0.378 > 0.322
    - It threats missing values as negatives

Collaborative Filtering

□ Similarity Metric

	I1	I2	I3	I4	I5	I6	I7
A	2/3	0	0	5/3	-7/3	0	0
B	1/3	1/3	-2/3	0	0	0	0
C	0	0	0	-5/3	1/3	4/3	0

- Intuitively we want: $sim(A, B) > sim(A, C)$
  - Jaccard similarity produces: 1/5 < 2/4
  - Cosine similarity produces: 0.378 > 0.322
    - It threats missing values as negatives
  - Pearson similarity produces: 0.092 > - 0.559
    - It subtracts row mean prior to computing cosine

Collaborative Filtering

- **Rating Predictions**
  - Let vector r_x represent user X's ratings
  - Let N be the set of k most similar users that rated item i
  - Prediction of X's ratings on item i
    - $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
      - Threats each similar user contribution equally
    - $r_{xi} = \frac{\sum_{y \in N} sim_{xy} \cdot r_{yi}}{\sum_{y \in N} sim_{xy}}$
      - Enables more similar users to contribute with greater weight
    - There are other options

Collaborative Filtering

- **Item-Item Collaborative Filtering**
  - So far, we considered User-User CF
  - Another perspective: Item-Item
    - For item i, find set of similar items N
    - Estimate rating for item i, based on ratings for similar items
    - Similarity measure and prediction calculations remains the same as in user-user model

$$r_{xi} = \frac{\sum_{j \in N} sim_{ij} \cdot r_{xj}}{\sum_{j \in N} sim_{ij}}$$

sim_{ij} is a similarity of items i and j
r_{xj} is a rating of user x on item j
N is a set of most similar items to i, rated by x

Collaborative Filtering

- **Item-Item CF (k = 2)**

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Missing ratings

Ratings [1, 5]

Collaborative Filtering

- **Item-Item CF (k = 2)**

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Estimate Rating [1, 5]

Collaborative Filtering

□ Item-Item CF (k = 2)

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

PCC as a similarity measure:

1.) Subtract row mean m_i from each row

$$m_1 = (1 + 3 + 5 + 5 + 4) / 5 = 3.6$$

$$row_1 = [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4]$$

2.) Compute Cosine similarity between rows

Collaborative Filtering

- **Item-Item CF (k = 2)**

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	Sim(1, m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		**0.59**

PCC as a similarity measure:

1.) Subtract row mean m_i from each row

$$m_1 = (1 + 3 + 5 + 5 + 4) / 5 = 3.6$$

$$row_1 = [-2.6, 0. -0.6, 0. 0. 1.4, 0, 0, 1.4]$$

2.) Compute Cosine similarity between rows

Collaborative Filtering

- **Item-Item CF (k = 2)**

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	Sim(1, m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		**0.59**

Predict missing value by using weighted average:

r1,5 = (0.41 * 2 + 0.59 * 3) / (0.41 + 0.59) = 2.6

$$r_{xi} = \frac{\sum_{y \in N} sim_{xy} \cdot r_{yi}}{\sum_{y \in N} sim_{xy}}$$

CF: Common Practice

- Estimate r_{xi} as the weighted average according to the following equation:

$$r_{xi} = \boxed{b_{xi}} + \frac{\sum_{j \in N} sim_{ij} \cdot (r_{xi} - b_{xj})}{\sum_{j \in N} sim_{ij}}$$

Baseline estimation for r_{xi} :

$$b_{xi} = \mu + b_x + b_i$$

μ is an overall mean movie rating

b_x is a rating deviation for user x

$$b_x = \overline{b_x} - \mu$$

b_i is a rating deviation for item i

$$b_i = \overline{b_i} - \mu$$

Collaborative Filtering

- **Item-Item vs. User-User**

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4

  - In practice: Item-Item works better!
  - Items are simpler, users have very different preferences

Collaborative Filtering

- **Hybrid Methods**
  - Implement two or more different recommender systems
    - Estimate predictions by incorporating output from more recommenders
    - For instance, compute the final prediction as a linear combination
    - $P = (1 - \lambda) \cdot UU + \lambda \cdot II, \lambda \in [0, 1]$
  - Add content based approach features to CF
    - Introduce item profiles to address new item issue
    - Include demographic features to deal with new user issue

Collaborative Filtering

- **Pros**
  - Works for various types of items
    - Universal approach, domain-specific knowledge not needed
- **Cons**
  - Cold Start
    - It needs users already in the system to provide predictions
  - Sparsity
    - Utility matrix is usually very sparse
    - Hard to find users that have rated same items
  - First Rater
    - It cannot recommend unrated item
    - New items, unpopular items
  - Popularity Bias
    - Black sheep problem (users with unique taste)
    - Favors popular items

Remarks, Practice & Advices

- **Evaluation**

users

movies

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

Remarks, Practice & Advices

- **Evaluation**

users

movies

1	2		3			3	
		3		3			
				2		1	
		3					
	2	4			?		?
				?	?	?	
		4					?

Testing Data Set

Remarks, Practice & Advices

- **Evaluating Predictions**
  - Compare predictions with known ratings
    - Root Mean Square Error (RMSE)
      - $\sqrt{\frac{1}{N}\sum_{x_i}\left(\widehat{r_{x_i}} - r_{x_i}\right)^2}$, where r_{x_i} is predicted and $\widehat{r_{x_i}}$ is the actual rating
    - Precision at top 10
      - % of those in top 10
    - Rank Correlation
      - Spearman's correlation between system's and user's rankings

Remarks, Practice & Advices

- **Evaluating Predictions**
  - 0/1 model
    - Coverage
      - Number of items/users for which the system can make predictions
    - Precision
      - Accuracy of predictions
    - Receiver Operating Characteristic (ROC)
      - Tradeoff curve between true positives and false positives

Remarks, Practice & Advices

- **Issues with Error Measures**
  - Focusing on accuracy may miss the point
    - Prediction Diversity
    - Prediction Context
    - Order of predictions
  - In recommenders good performance for high ratings is important
    - RMSE emphases errors, method that performs well for high ratings and bad for other might get penalized

Remarks, Practice & Advices

- **CF: Computational Complexity**
  - Expensive step: finding top k most similar users
    - $o(N \cdot M)$, N is a number of users, M is a number of items
  - Too expensive in runtime, solution is to precompute
    - For instance for User-User CF
    - $o(N^2 \cdot M)$
    - Obviously, it will not scale…
  - There is a better solution
    - Large scale neighbor identification
      - LSH, SimHash, MinHash
    - Clustering – reduces the space of potential solutions

Remarks, Practice & Advices

- **Tip: Add more data**
  - All the data should be utilized
    - No use in data reduction to make fancy algorithms work
    - Simple methods on large data sets do best
  - Add even more data
    - E.g. for movies add IMDB data about genres
  - More data outperforms fancy algorithms
    - http://anand.typepad.com/datawocky/2008/03/more-data-usual.html