Mining Massive Datasets

Lecture 5

Artur Andrzejak
http://pvs.ifi.uni-heidelberg.de

RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

Note on Slides

A substantial part of these slides come (either verbatim or in a modified form) from the book
Mining of Massive Datasets
by Jure Leskovec, Anand Rajaraman, Jeff Ullman
(Stanford University).
For more information, see the website accompanying the book: http://www.mmds.org.

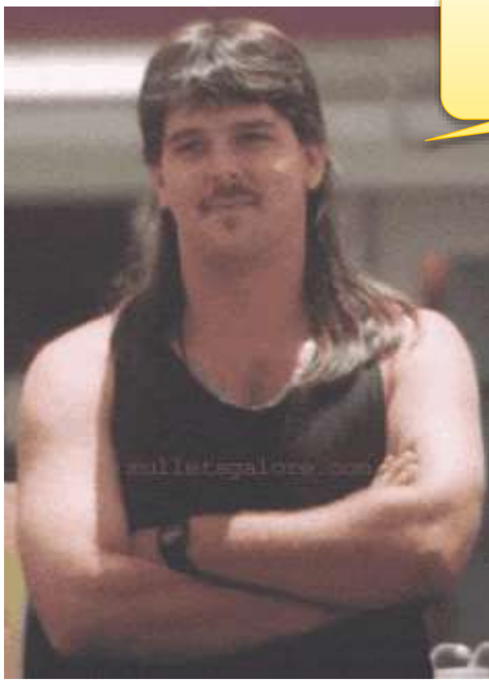
Current Topic

High dim. data	Graph data	Infinite data	Machine learning	Apps
Locality sensitive hashing	PageRank, SimRank	Filtering data streams	SVM	Recommender systems
Clustering	Community Detection	Web advertising	Decision Trees	Association Rules
Dimensio-nality reduction	Spam Detection	Queries on streams	Perceptron, kNN	Duplicate document detection

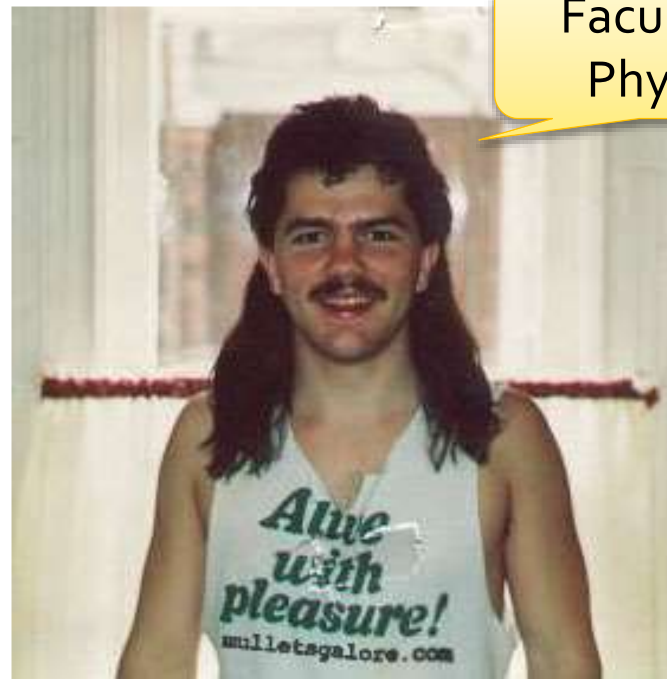
Programming in Spark & MapReduce

Recommender Systems: Content-based Systems & Collaborative Filtering

Example: Recommender Systems



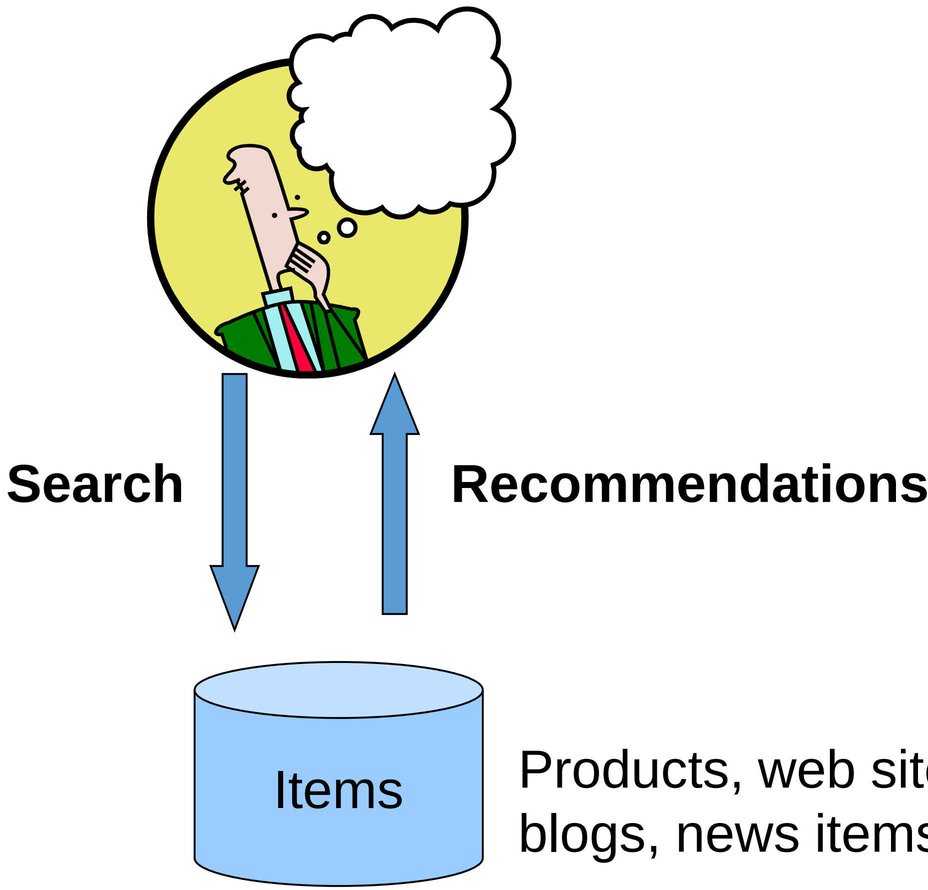
Faculty of Math & CS



Faculty of Physics

- **Scholar X**
  - Buys Metallica CD
  - Buys Megadeth CD
- **Scholar Y**
  - Does search on Metallica
  - Recommender system suggests Megadeth from data collected about customer **X**

Recommendations



Products, web sites, blogs, news items, …

Examples:

amazon.com

PANDORA

StumbleUpon

del.icio.us

NETFLIX

movielens
helping you find the *right* movies

last.fm
the social music revolution

Google News

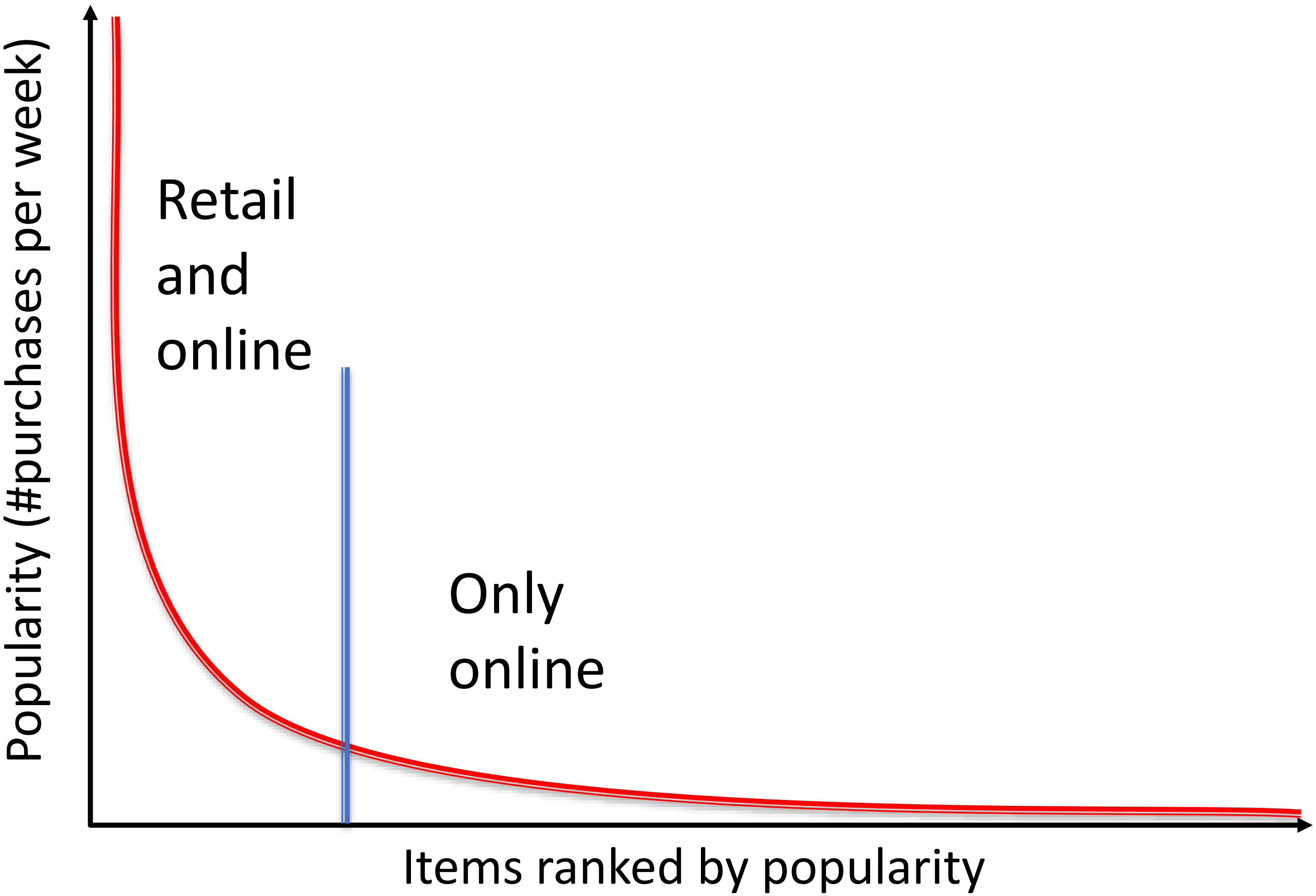
YouTube

XBOX LIVE

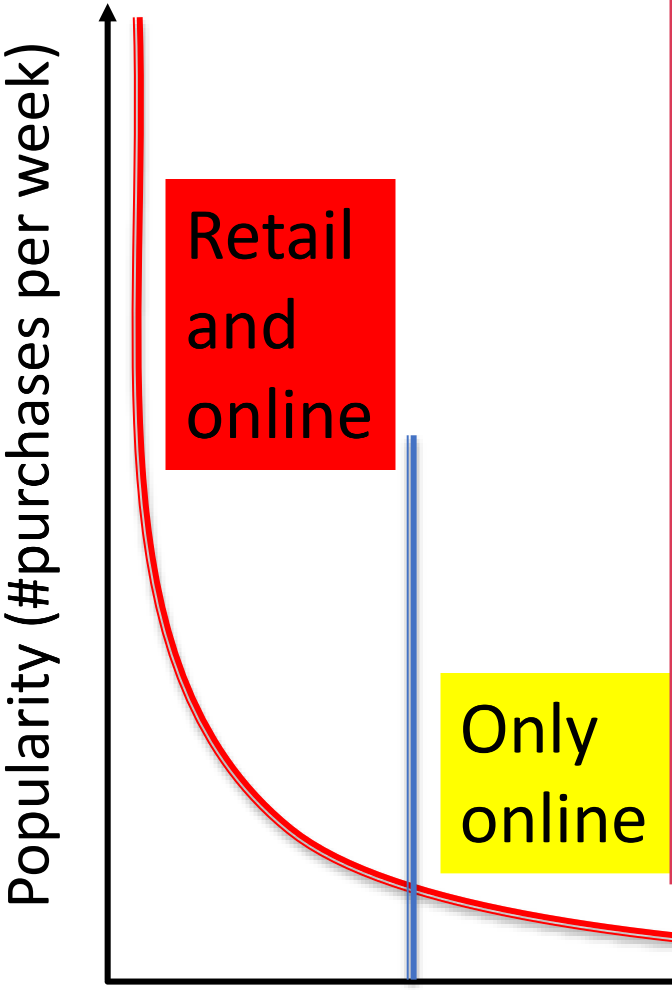
From Scarcity to Abundance

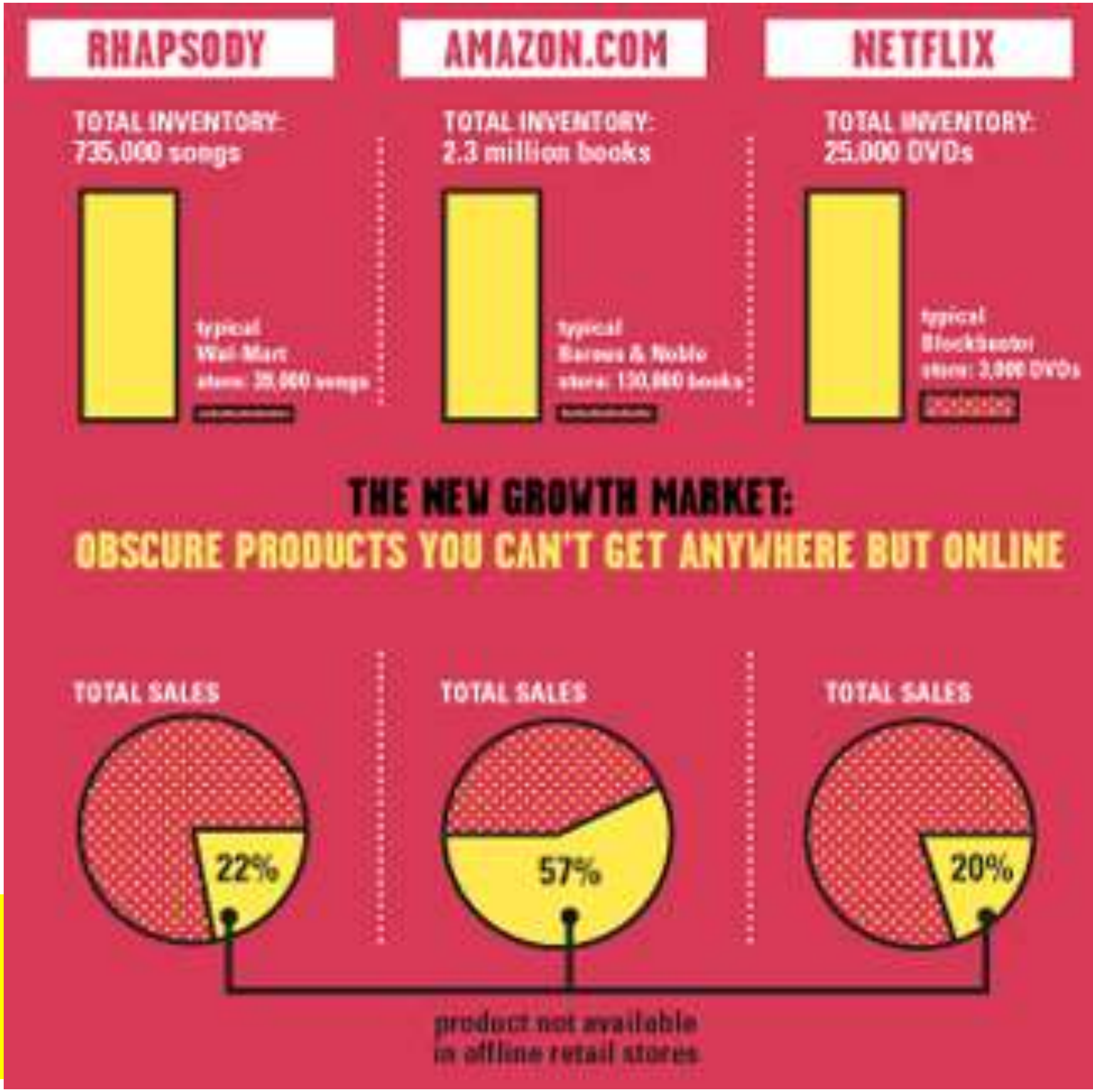
- **Shelf space is a scarce commodity for traditional retailers**
  - Also: TV networks, movie theaters,…
- **Web enables near-zero-cost dissemination of information about products**
  - From scarcity to abundance
- More choice necessitates **better filters**
  - Recommendation engines
  - How **Into Thin Air** made **Touching the Void** a bestseller: http://www.wired.com/wired/archive/12.10/tail.html

The Long Tail



The Long Tail





Formal Model

- $\boldsymbol{X}$ = set of **Customers**
- $\boldsymbol{S}$ = set of **Items**

- **Utility function** $\boldsymbol{u}: \boldsymbol{X} \times \boldsymbol{S} \rightarrow \boldsymbol{R}$
  - $\boldsymbol{R}$ = set of ratings
  - $\boldsymbol{R}$ is a totally ordered set
  - e.g., **0-5** stars, real number in **[0,1]**

Utility Matrix

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

Key Problems

- (1) **Gathering "known" ratings for matrix**
  - How to collect the data in the utility matrix
- (2) **Extrapolate unknown ratings from the known ones**
  - Mainly interested in high unknown ratings
    - We are not interested in knowing what you don't like but what you like
- (3) **Evaluating extrapolation methods**
  - How to measure success/performance of recommendation methods

(1) Gathering Ratings

- **Explicit**
  - Ask people to rate items
  - Doesn't work well in practice – people can't be bothered
- **Implicit**
  - Learn ratings from user actions
    - E.g., purchase implies high rating
  - What about low ratings?

(2) Extrapolating Utilities

- **Key problem:** Utility matrix ***U*** is <u>sparse</u>
  - Most people have not rated most items
  - **Cold start:**
    - New items have no ratings
    - New users have no history
- Three approaches to recommender systems:
  - **1) Content-based**
  - **2) Collaborative**
  - **3)** Latent factor based

Today

Content-based Recommender Systems

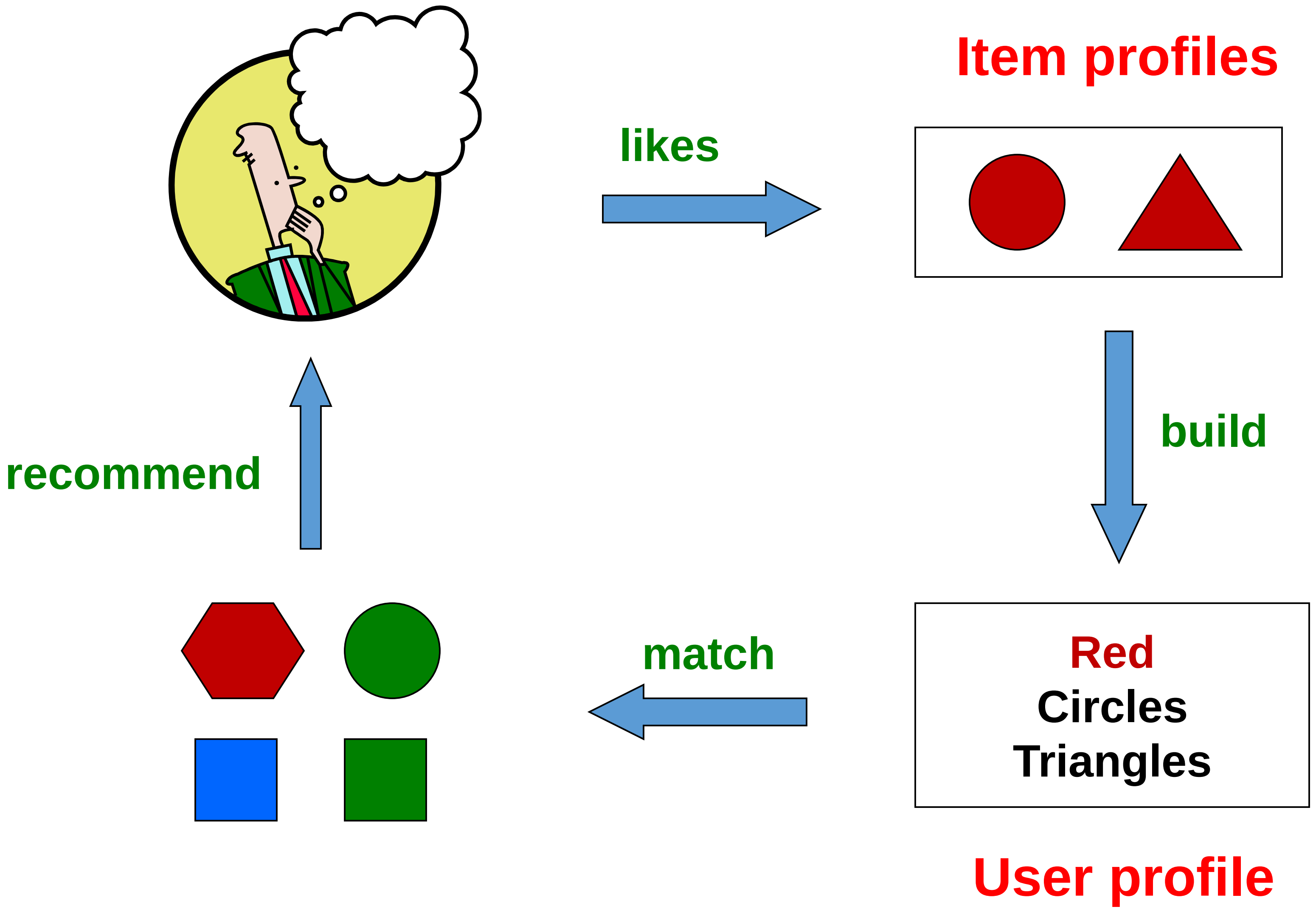
Content-based Recommendations

- **Main idea:** Recommend to customer ***x*** items similar to previous items rated highly by ***x***

Example:

- **Movie recommendations**
  - Recommend movies with same actor(s), director, genre, …
- **Websites, blogs, news**
  - Recommend other sites with "similar" content

Plan of Action



Item Profiles

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

Sidenote: TF-IDF

- Let f_{ij} be frequency of term i in document j
- Then:

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

Note: we normalize TF to discount for "longer" docs

- Let n_i = number of docs that mention term i
- And N = total number of docs
- Then:

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

The **TF-IDF score** is (term i, doc j): $w_{ij} = TF_{ij} \times IDF_i$

⇒ Use this to define a (variant of) **doc profile:**

= Set of words with highest **TF-IDF** scores (together with these scores)

Example: Items are Movies

- Representing item profile – a “mixed” vector

$$i_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ \vdots \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 3.2 \\ 2.9 \\ \vdots \end{bmatrix}$$

Set of actors (as 0/1 entries)

Director(s) (as 0/1 entries)

Ratings from various movie DBs (as numbers)

User Profiles – 1st Atte

In general, does not work!

- Intuition: average the profiles of all items rated by a user and weight them by the ratings of this user
- User u gives items 1, 2, ..., n ratings $r_1, r_2, \ldots, r_n$
- **User profile x**
  - **x** = weighted average of rated item profiles

$$\mathbf{x} = (r_1 * i_1 + r_2 * i_2 + \ldots + r_n * i_n)/n$$

Weights = ratings

Profiles of items 1, 2, .., n (those rated by user u) - vectors

Example: Star-Based Ratings /1

$$i_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ i_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ i_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},\ i_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},\ i_5 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

← actor A present

← actor B present

- Items are movies, only features are "actors"
  - Item profile has 2 components (for actor A and actor B)
- User ratings are 1 to 5 stars (per movie)
- User watched 5 movies
  - Actor A – movies got 3 and 5 stars (movies 1 & 2)
  - Actor B – movies got 1, 2 and 4 stars (movies 3, 4, 5)
- Ratings are r_1=3, r_2=5, r_3=1, r_4=2, r_5=4
- Item profiles are as above

Example: Star-Based Ratings /2

- The user profile becomes:

$$(r_1 i_1 + r_2 i_2 + r_3 i_3 + r_4 i_4 + r_5 i_5)/5 =$$

$$(3\begin{bmatrix}1\\0\end{bmatrix} + 5\begin{bmatrix}1\\0\end{bmatrix} + 1\begin{bmatrix}0\\1\end{bmatrix} + 2\begin{bmatrix}0\\1\end{bmatrix} + 4\begin{bmatrix}0\\1\end{bmatrix})/5 = \begin{bmatrix}8/5\\7/5\end{bmatrix}$$

- Problem 1: user likes actor A more than actor B, but this shows only weakly in his profile!
- Problem 2: with more ratings, each component becomes smaller (as **n** gets larger)
  - Because components with value 0 disturb the average, but should be treated as “don’t care about corresp. rating”

For 1: Normalizing Ratings

- Solution for 1: Normalize ratings by subtracting user's mean rating (which is 3 = (3+5+1+2+4)/5)
  - Normalized ratings for actor A movies => 0, +2
  - Normalized ratings for actor A movies => -2, -1, +1
- Then the user profile is:
  - With r_1=0, r_2=+2, r_3=-2, r_4=-1, r_5=+1

$$(0\begin{bmatrix}1\\0\end{bmatrix}+2\begin{bmatrix}1\\0\end{bmatrix}-2\begin{bmatrix}0\\1\end{bmatrix}-1\begin{bmatrix}0\\1\end{bmatrix}+1\begin{bmatrix}0\\1\end{bmatrix})/5=\begin{bmatrix}2/5\\-2/5\end{bmatrix}$$

Now better: clear distinction for actor A and actor B

For 2: Per-component Weights

Only ratings for these 2 items should be counted for actor A

$$(0\begin{bmatrix}1\\0\end{bmatrix}+2\begin{bmatrix}1\\0\end{bmatrix}-2\begin{bmatrix}0\\1\end{bmatrix}-1\begin{bmatrix}0\\1\end{bmatrix}+1\begin{bmatrix}0\\1\end{bmatrix})/5=\begin{bmatrix}2/5\\-2/5\end{bmatrix}$$

Only the ratings for these 3 items should be counted for actor B

- Essence of problem 2: a 0 in an item's component (=attribute) k should mean "don't care", but now mean "one more neutral rating for attribute k"
- => Use "individual" n for each vector component
  - For actor A: $n_A = 2$, for actor B: $n_B = 3$

For 2: Per-component Weights

- => Use "individual" n for each vector component
  - For actor A: $n_A = 2$, for actor B: $n_B = 3$
  - Recall: Normalized ratings are $r_1=0$, $r_2=+2$, $r_3=-2$, $r_4=-1$, $r_5=+1$
- Then the user profile becomes:

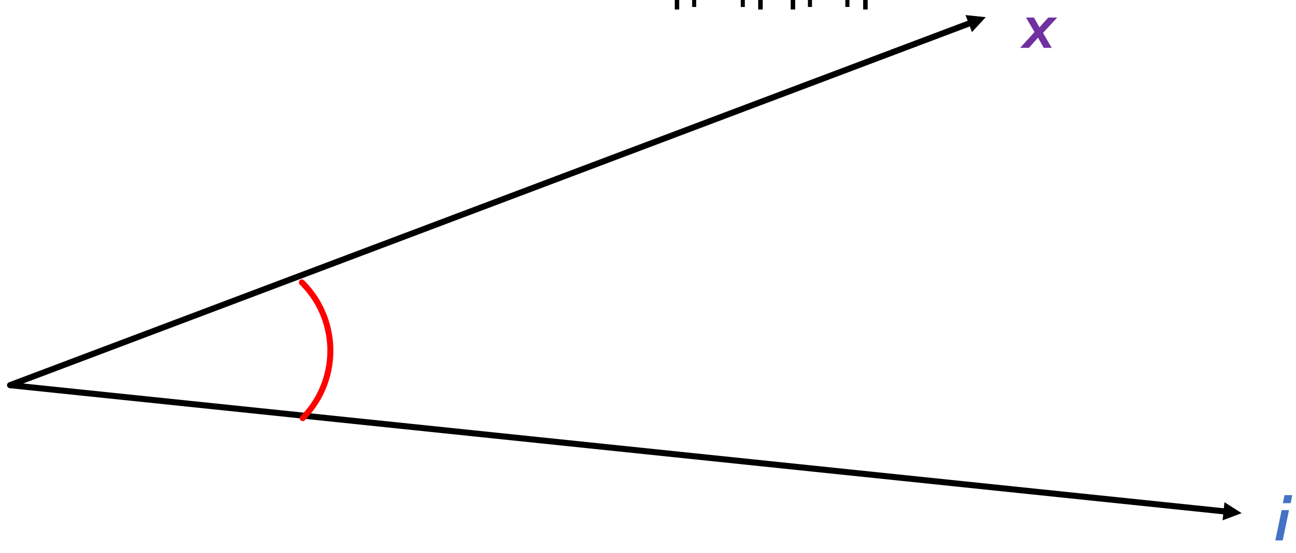
$$\begin{bmatrix} r_1/n_a \\ 0 \end{bmatrix} + \begin{bmatrix} r_2/n_a \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ r_3/n_b \end{bmatrix} + \begin{bmatrix} 0 \\ r_4/n_b \end{bmatrix} + \begin{bmatrix} 0 \\ r_5/n_b \end{bmatrix} =$$

$$\begin{bmatrix} 0/2 \\ 0 \end{bmatrix} + \begin{bmatrix} 2/2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -2/3 \end{bmatrix} + \begin{bmatrix} 0 \\ -1/3 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 1 \\ -2/3 \end{bmatrix}$$

Matching User and Item Profiles

- To compute similarity of user profile and item profile, use a prediction heuristic:
  - Given user profile $\boldsymbol{x}$ and item profile $\boldsymbol{i}$, estimate

$$\boldsymbol{u}(\boldsymbol{x},\boldsymbol{i}) = \cos(\boldsymbol{x},\boldsymbol{i}) = \frac{\boldsymbol{x}\cdot\boldsymbol{i}}{||\boldsymbol{x}||\cdot||\boldsymbol{i}||}$$

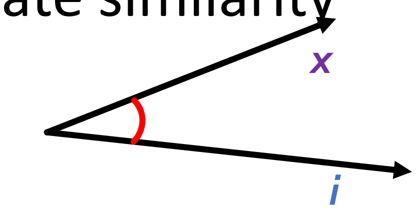


Summary: Content-Based R.

- We construct a vector $\boldsymbol{i}$ **for each item** (“**item profile**”) and a vector $\boldsymbol{x}$ (of size s) **for each user**
  - Item profile $\boldsymbol{i}$: “natural” attributes of an item
  - User vector $\boldsymbol{x}$: combination of item profiles rated by this user

- **Prediction heuristic**:
  - Given a user vector $\boldsymbol{x}$ and item vector $\boldsymbol{i}$, estimate similarity

$$\boldsymbol{u}(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$$



  - For a user with vector $\boldsymbol{x}$, recommend by various criteria:
    - E.g. all items with $u(\boldsymbol{x}, \boldsymbol{i})$ > **threshold**
    - Rank items by $u(\boldsymbol{x}, \boldsymbol{i})$, recommend top k (e.g. k=5)

Pros: Content-based Approach

- **+: No need for data on other users**
  - No cold-start or sparsity problems
- **+: Able to recommend to users with unique tastes**
- **+: Able to recommend new & unpopular items**
  - No first-rater problem
- **+: Able to provide explanations**
  - Can provide explanations of recommended items by listing content-features that caused an item to be recommended

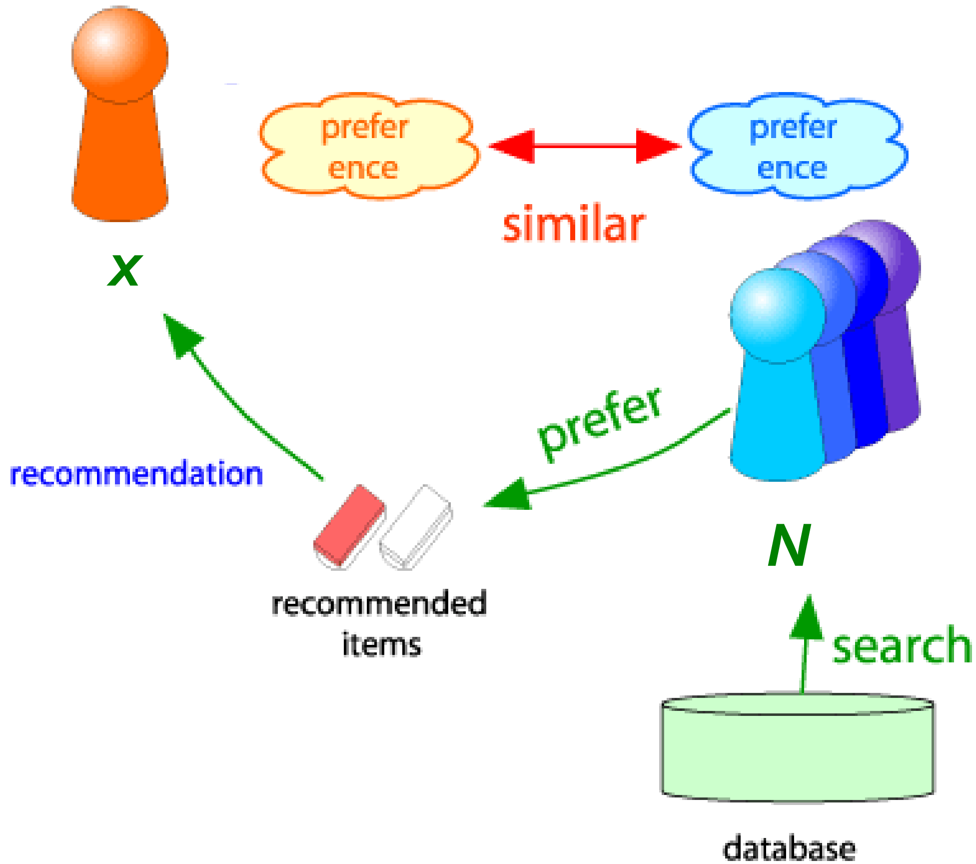
Cons: Content-based Approach

- **–: Finding the appropriate features is hard**
  - E.g., images, movies, music
- **–: Recommendations for new users**
  - **How to build a user profile?**
- **–: Overspecialization**
  - Never recommends items outside user's content profile
  - People might have multiple interests
  - **Unable to exploit quality judgments of other users**

Collaborative Filtering

User-User Collaborative Filtering

- Consider user ***x***
- Find set ***N*** of other users whose ratings are "**similar**" to ***x***'s ratings
- Estimate ***x***'s ratings based on ratings of users in ***N***



Finding "Similar" Users

- Let $\boldsymbol{r_x}$ be the vector of user $\boldsymbol{x}$'s ratings

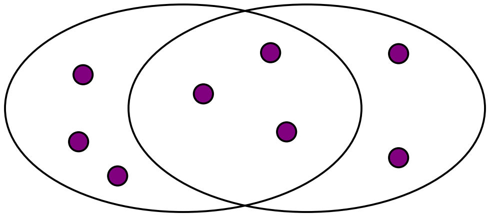
$$\boldsymbol{r_x} = [*, \_, \_, *, ***]$$
$$\boldsymbol{r_y} = [*, \_, **, **, \_]$$

- **Cosine similarity measure**
  - sim($\boldsymbol{x}$, $\boldsymbol{y}$) = cos($\boldsymbol{r_x}$, $\boldsymbol{r_y}$) = $\frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$
  - **Problem:** Treats missing ratings as "negative"

r_x, r_y as points:
$\boldsymbol{r_x}$ = {1, 0, 0, 1, 3}
$\boldsymbol{r_y}$ = {1, 0, 2, 2, 0}

Jaccard Measures

- The **Jaccard similarity** of two sets is the size of their intersection divided by the size of their union:
  $sim(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$
- **Jaccard distance:** $d(C_1, C_2) = 1 - |C_1 \cap C_2| / |C_1 \cup C_2|$



3 in intersection
8 in union
Jaccard similarity = 3/8
Jaccard distance = 5/8

- For measuring similarity of users, we consider <u>only sets of items</u> for which users voted
- Problem? Values of ratings are ignored!

r_x = [*, _, _, *, ***]
r_y = [*, _, **, **, _]

r_x, r_y as sets:
$r_x = \{1, 4, 5\}$
$r_y = \{1, 3, 4\}$

A Good Similarity Metric

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- Intuitively we want: **sim(*A*, *B*) > sim(*A*, *C*)**
  - **Jaccard similarity:** 1/5 **<** 2/4 => bad
  - **Cosine similarity:** 0.386 **>** 0.322 => not good
- **Solution: subtract the (row) mean**

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	$2/3$			$5/3$	$-7/3$		
B	$1/3$	$1/3$	$-2/3$				
C				$-5/3$	$1/3$	$4/3$	
D		0					0

Notice: cos similarity is = **correlation** when data is centered at 0!

sim A,B vs. A,C:

0.092 > -0.559

=> $sim(x, y) = \frac{\sum_i r_{xi} \cdot r_{yi}}{\sqrt{\sum_i r_{xi}^2} \cdot \sqrt{\sum_i r_{yi}^2}}$

From Cosine to Pearson

- Let $\boldsymbol{r_x}$ be the vector of user ***x***'s ratings
- **Cosine similarity measure**
  - sim(***x***, ***y***) = cos($\boldsymbol{r_x}$, $\boldsymbol{r_y}$) = $\dfrac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$
- **Pearson correlation coefficient**
  - S_{xy} = items rated by both users ***x*** and ***y***

$\overline{\mathbf{r}_x}$, $\overline{\mathbf{r}_y}$ … avg. rating of **x**, **y**

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

$\boldsymbol{r_x}$ = [*, _, _, *, ***]
$\boldsymbol{r_y}$ = [*, _, **, **, _]

Rating Predictions

From similarity metric to recommendations:

- Let $\boldsymbol{r_x}$ be the vector of user ***x***'s ratings
- Let ***N*** be the set of ***k*** users most similar to ***x*** (according to sim(***x***, ***y***)) who have rated item ***i***
- Prediction for item ***i*** of user ***x***:
  - $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
  - $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$

Users rating of item i := avg. rating of k most similar users

Better: weights a recommendation of other user y by similarity s_{xy} to this "neighbor" y

Shorthand:
$\boldsymbol{s_{xy} = sim(x, y)}$

Item-Item Collaborative Filtering

- **Another view: Item-item**
  - For item ***i***, find other similar items rated by user x
  - Estimate rating for item ***i*** based on ratings for similar items (with ratings by user x)
    - Note: We can use same similarity metrics and prediction functions as in user-user model

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

Shorthand:
$s_{ij} = sim(i, j)$

r_{xj} rating of user ***x*** on item ***j***
N(i;x) set of items rated by ***x*** similar to ***i***

Example: Item-Item CF (|N|=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	

- unknown rating

- rating between 1 to 5

Item-Item CF (|N|=2)

■ - estimate rating of movie **1** by user **5**

users / **movies**

movies \ users	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	

Item-Item CF (|N|=2)

Neighbor selection:
Identify movies similar to movie **1**, rated by user 5

We use Pearson correlation as similarity:
1) Subtract mean rating m_i from each movie i
$m_1 = (1+3+5+5+4)/5 = \mathbf{3.6}$
row 1: *[-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]*
2) Compute cosine similarities between rows

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	
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**

Item-Item CF (|N|=2)

Compute similarity weights:

$$s_{1,3} = 0.41,\ s_{1,6} = 0.59$$

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	-
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

Item-Item CF (|N|=2)

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

Predict by taking weighted average:

$r_{1,5} = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6$

users

movies

	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		2.6	5			5		4		1.00
2			5	4			4			2	1	3	
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

CF: Common Practice

Before (simpler):

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

- Define similarity s_{ij} of items ***i*** and ***j***
- Select ***k*** nearest neighbors ***N(i; x)***
  - Items most similar to ***i***, that were rated by ***x***
- Estimate rating r_{xi} as the weighted average:

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

baseline estimate for (*x*,*j*)

baseline estimate for (*x*,*i*):

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

- μ = overall mean item rating
- b_x = rating deviation of user ***x***
  = *(avg. rating of user* ***x****)* – μ
- b_i = rating deviation of item ***i***

Item-Item vs. User-User

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.8	
Bob		0.5		0.3
Carol	0.9		1	0.8
David			1	0.4

- In practice, it has been observed that item-item often works better than user-user
- **Why?** Items are simpler, users have multiple tastes

Pros/Cons of Collaborative Filtering

- **+ Works for any kind of item**
  - No feature selection needed
- **- Cold Start:**
  - Need enough users in the system to find a match
- **- Sparsity:**
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items
- **- First rater:**
  - Cannot recommend an item that has not been previously rated (e.g. new items, esoteric items)
- **- Popularity bias:**
  - Cannot recommend items to someone with unique taste
  - Tends to recommend popular items

Hybrid Methods

- Implement two or more different recommenders and combine predictions
  - Perhaps using a linear model
- Add content-based methods to collaborative filtering
  - Item profiles for new item problem
  - Demographics to deal with new user problem

Remarks & Practical Tips

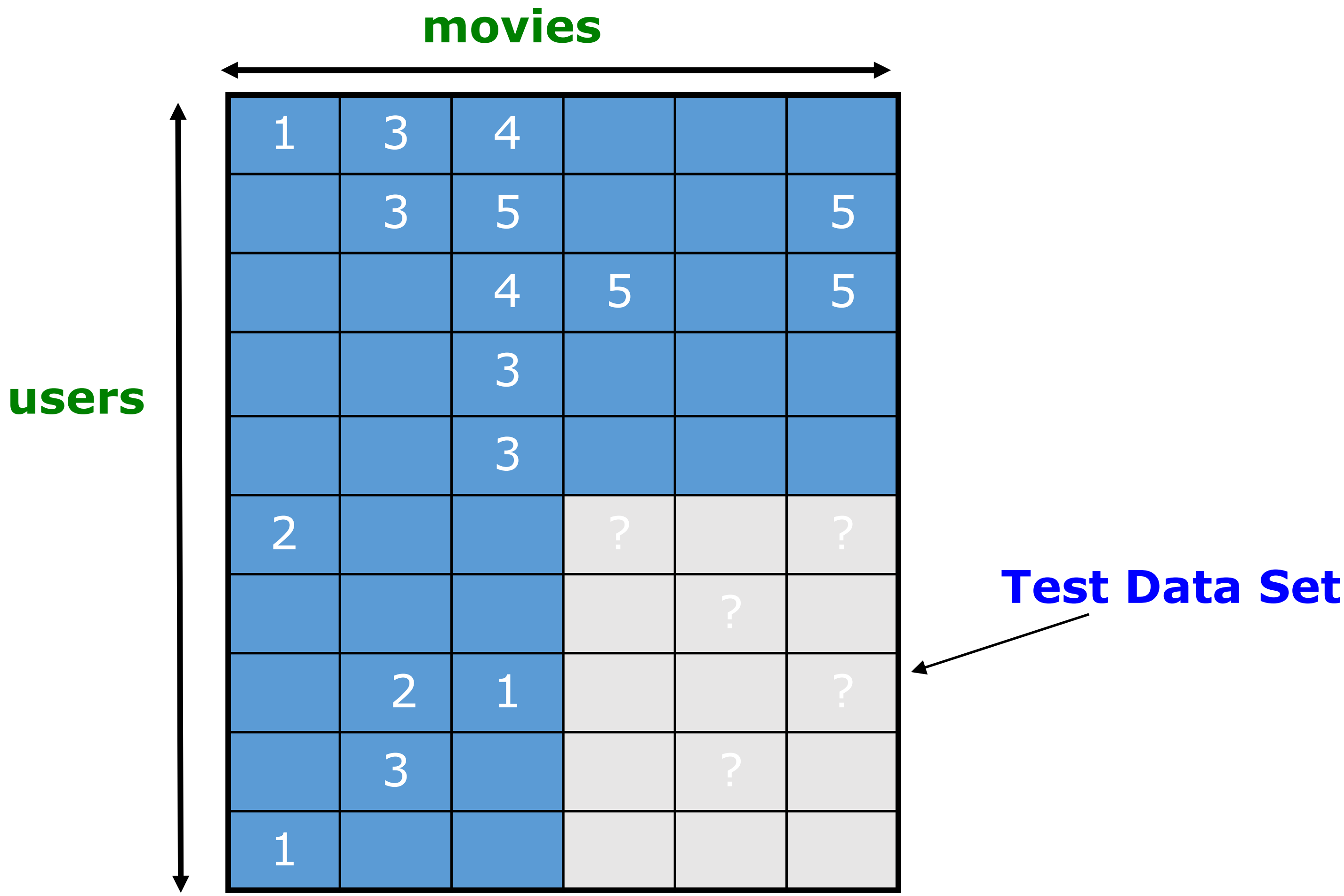
Evaluation

movies

users

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

Evaluation



Evaluating Predictions

- **How to compare predictions with known ratings?**
  - **Root-mean-square error** (RMSE), details: [link]
    - $\sqrt{\frac{1}{N}\sum_{xi}\left(r_{xi} - r_{xi}^{*}\right)^{2}}$
    - where $\boldsymbol{r_{xi}}$ is predicted, $\boldsymbol{r_{xi}^{*}}$ is the true rating of ***x*** on ***i***, and ***N*** is the number of ratings (= number of used ***(x,i)*** combinations)
  - **Precision at top 10**: % of those in top 10
- Another approach: **0/1 model**
  - **Coverage:**
    - Number of items/users for which system can make predictions
  - **Precision:**
    - Accuracy of predictions
  - **Receiver operating characteristic** (ROC)
    - Tradeoff curve between false positives and false negatives

Collaborative Filtering: Complexity

- Expensive step is finding ***k*** most similar customers: **$O(|X|)$**
- Too expensive to do at runtime
  - Could pre-compute
- Naïve pre-computation takes time **$O(k \cdot |X|)$**
    - X … set of customers
- We already know how to do this!
  - Near-neighbor search in high dimensions (**LSH**)
  - Clustering
  - Dimensionality reduction

Tip: Add Data

- **Leverage all the data**
  - Don’t try to reduce data size in an effort to make fancy algorithms work
  - Simple methods on large data do best
- **Add more data**
  - e.g., add IMDB data on genres
- **More data beats better algorithms**

http://anand.typepad.com/datawocky/2008/03/more-data-usual.html

Thank you.

Questions?