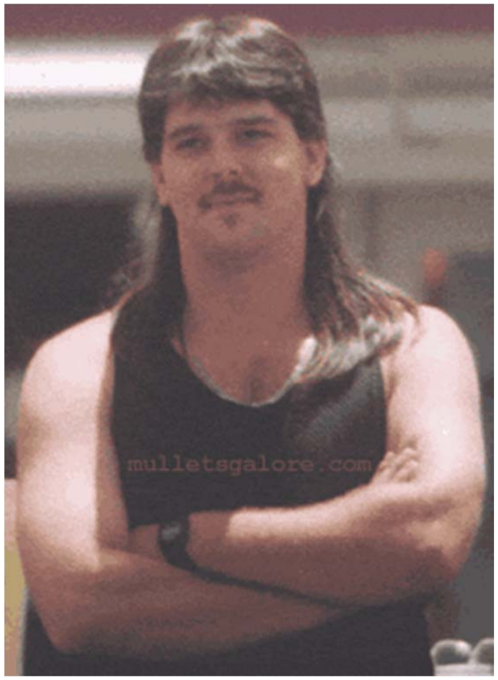
Recommender Systems: Content-based Systems & Collaborative Filtering

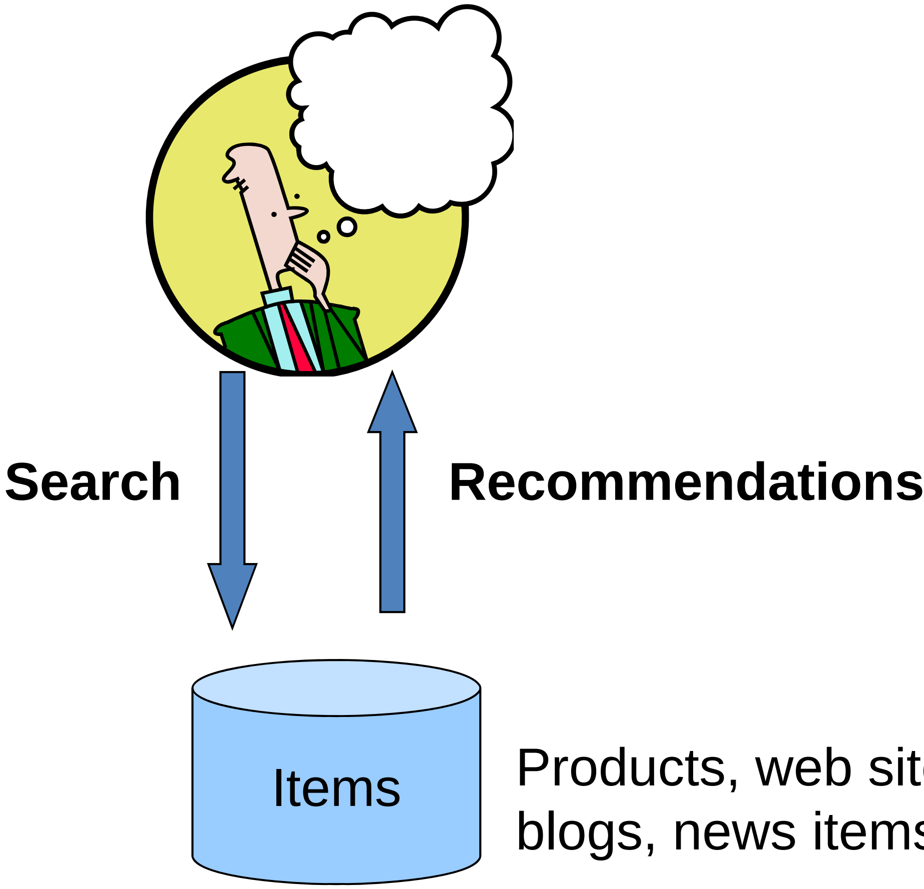
Example: Recommender Systems



- **Customer X**
  - Buys Metalica CD
  - Buys Megadeth CD

- **Customer Y**
  - Does search on Metalica
  - Recommender system suggests Megadeth from data collected from customer X

Recommendations



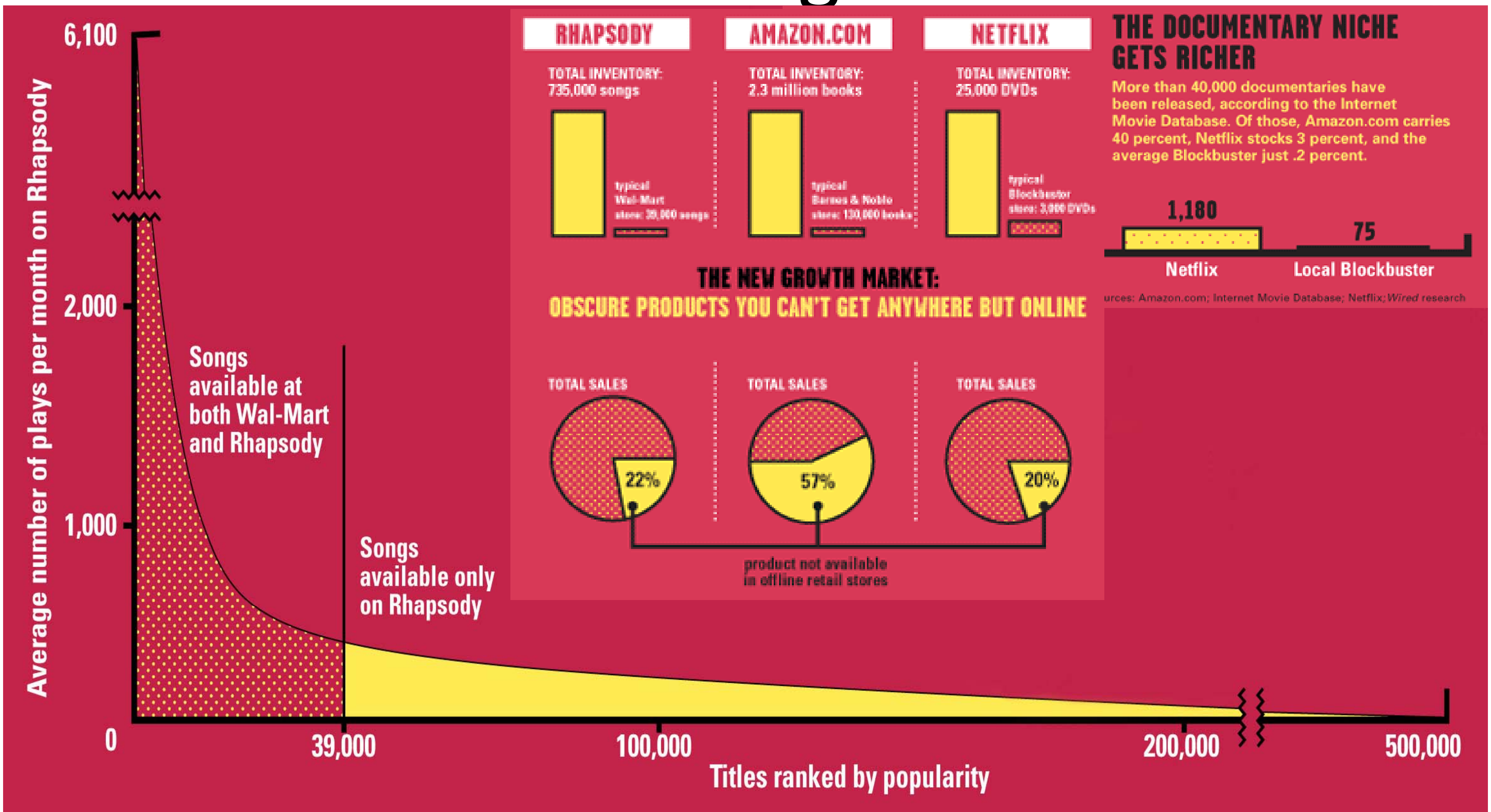
Products, web sites, blogs, news items, …

Examples:

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

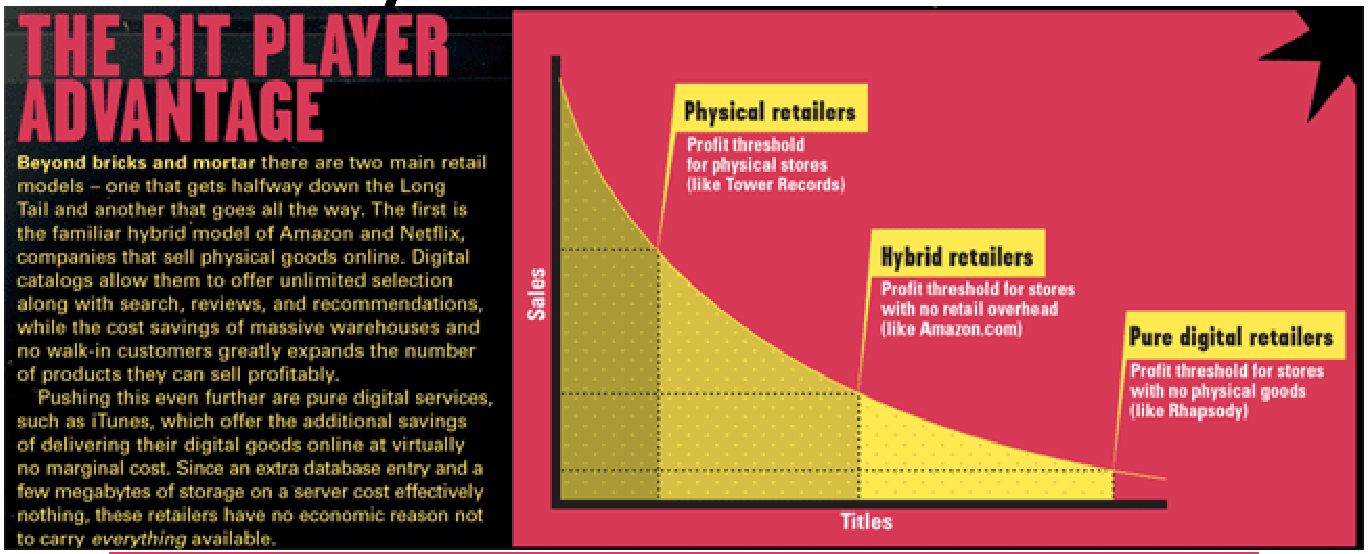
The Long Tail



Sources: Erik Brynjolfsson and Jeffrey Hu, MIT, and Michael Smith, Carnegie Mellon; Barnes & Noble; Netflix; RealNetworks

Source: Chris Anderson (2004)

Physical vs. Online



Read http://www.wired.com/wired/archive/12.10/tail.html to learn more!

Types of Recommendations

- **Editorial and hand curated**
  - List of favorites
  - Lists of “essential” items
- **Simple aggregates**
  - Top 10, Most Popular, Recent Uploads
- **Tailored to individual users**
  - Amazon, Netflix, …

Formal Model

- C = set of **Customers**
- S = set of **Items**

- **Utility function** u: $C \times S \rightarrow R$
  - R = set of ratings
  - 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

- **Gathering "known" ratings for matrix**
- **Extrapolate unknown ratings from known ratings**
  - Mainly interested in high unknown ratings
- **Evaluating extrapolation methods**

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?

Extrapolating Utilities

- **Key problem:** matrix U is sparse
  - Most people have not rated most items
  - **Cold start:**
    - New items have no ratings
    - New users have no history

- **Three approaches to Recommender Systems:**
  - Content-based
  - Collaborative
  - Hybrid

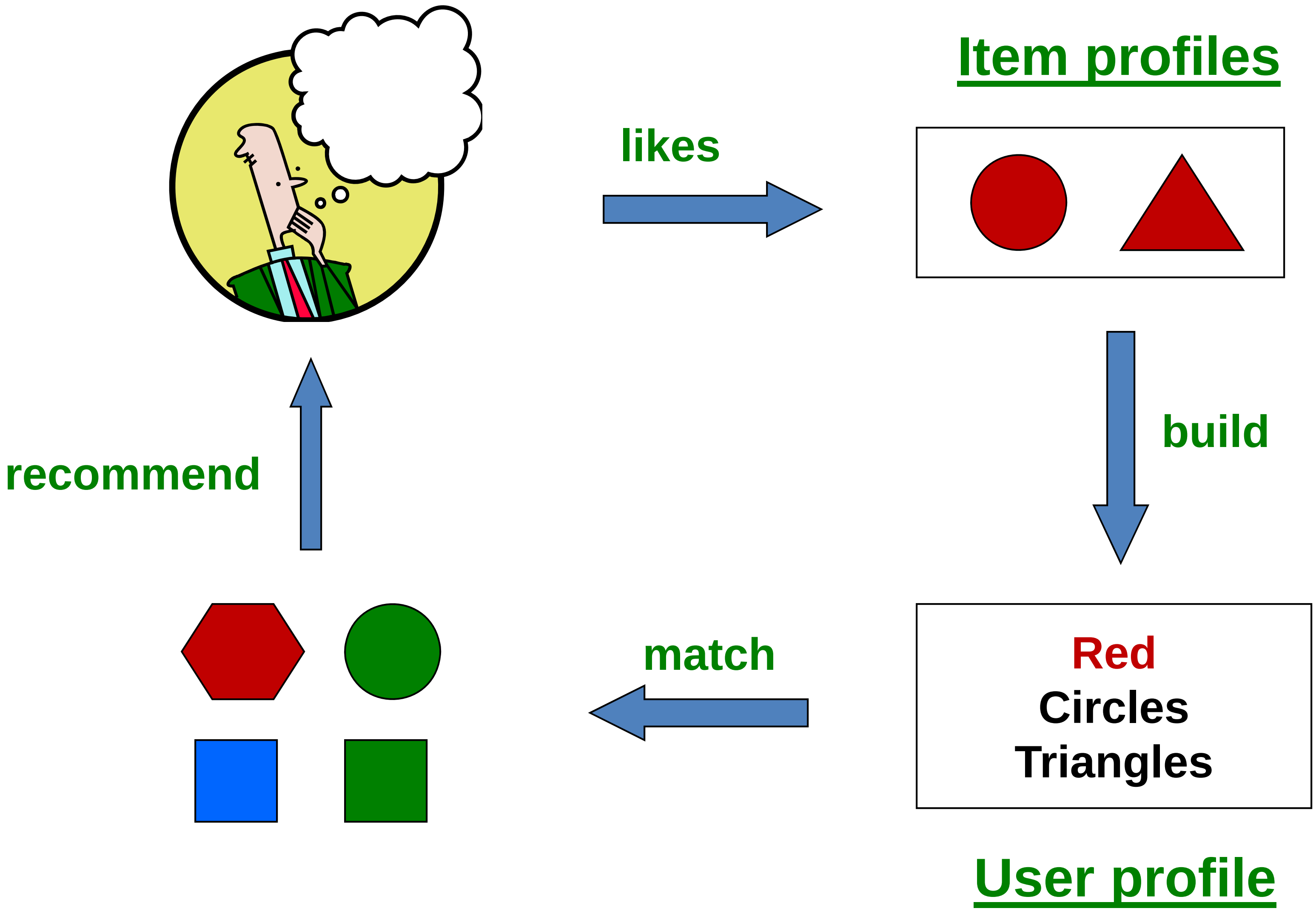
Content-based Recommendations

- **Main idea:** Recommend items to customer *x* 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
- **How to pick important features?**
  - Usual heuristic from text mining is TF-IDF (Term frequency * Inverse Doc Frequency)
    - Term … feature
    - Document … item

Sidenote: TF-IDF

f_{ij} = frequency of term (feature) i in document (item) j

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

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

n_i = number of docs that mention term i

N = total number of docs

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

TF-IDF score: $w_{ij} = TF_{ij} \times IDF_i$

Doc profile = set of words with highest TF-IDF scores, together with their scores

User Profiles and Prediction

- **User profile possibilities:**
  - Weighted average of rated item profiles
  - Variation: weight by difference from average rating for item
  - …
- **Prediction heuristic:**
  - Given user profile $\boldsymbol{u}$ and item profile $\boldsymbol{i}$, estimate $u(\boldsymbol{u},\boldsymbol{i}) = \cos(\boldsymbol{u},\boldsymbol{i}) = \boldsymbol{u}\cdot\boldsymbol{i}/(|\boldsymbol{u}||\boldsymbol{i}|)$
  - Need efficient method to find items with high utility: LSH!

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 and unpopular items**
  - No first-rater problem
- Can provide explanations of recommended items by listing content-features that caused an item to be recommended

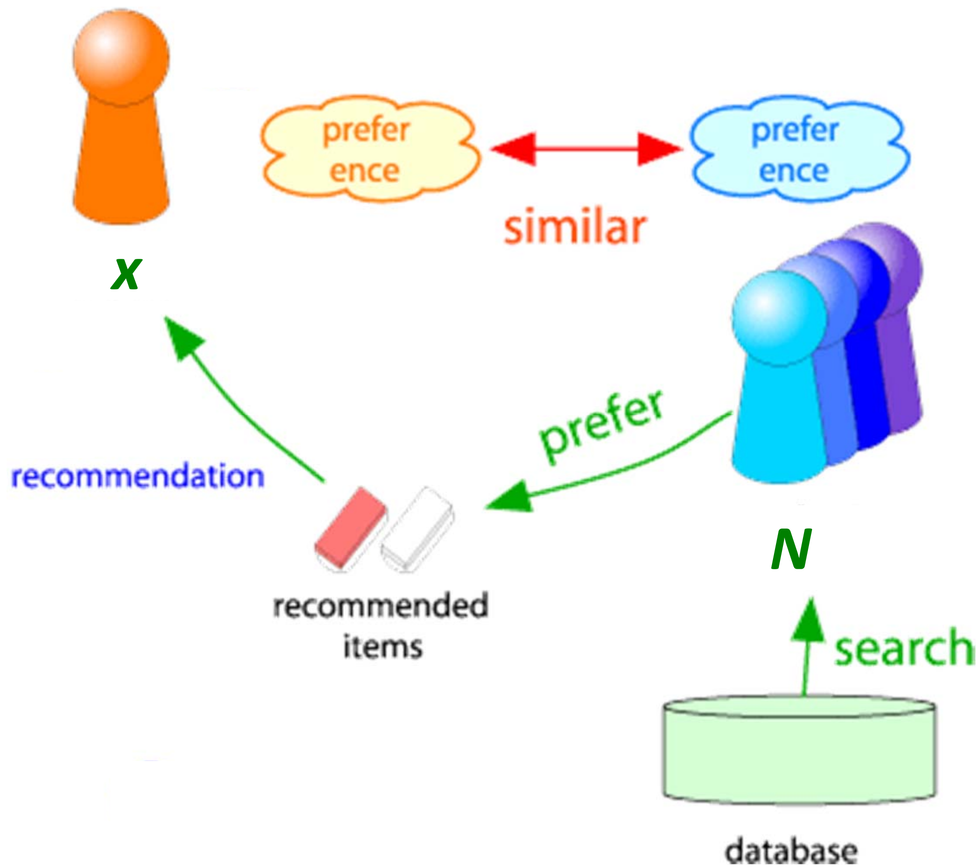
Cons: Content-based Approach

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

Collaborative Filtering

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*



Similar Users

- Let r_x be the vector of user x's ratings
- **Jaccard similarity measure**
  - Problem: Ignores the value of the rating
- **Cosine similarity measure**
  - sim(x,y) = cos(r_x , r_y)
  - Problem: Treats missing ratings as "negative"
- **Pearson correlation coefficient**
  - S_{xy} = items rated by both users x and y

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

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
- Cosine similarity: 0.386 **>** 0.322
  - Considers missing ratings as "negative"
  - **Solution:** subtract the 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

sim A,B vs. A,C:
0.092 **>** -0.559

Notice cos sim is correlation when data is centered at 0

Rating Predictions

- Let r_x be the vector of user *x*'s ratings
- Let *N* be the set of *k* users most similar to *x* who have rated item *i*
- **Possibilities for prediction for item *s* of *user x*:**
  - $r_{xi} = 1/k \sum_{y \in N} r_{yi}$
  - $r_{xi} = (\sum_{y \in N} \text{sim}(x,y)\ r_{yi}) \ / \ (\sum_{y \in N} \text{sim}(x,y))$
  - Other options?
- **Many tricks possible…**

Item-Item Collaborative Filtering

- **So far: User-user collaborative filtering**
- **Another view: Item-item**
  - For item *i*, find other similar items
  - Estimate rating for item based on ratings for similar items
  - Can use same similarity metrics and prediction functions as in user-user model

$$r_{ui} = \frac{\sum_{j \in N(i;u)} s_{ij} r_{uj}}{\sum_{j \in N(i;u)} s_{ij}}$$

s_{ij}… similarity of items *i* and *j*
r_{uj}…rating of user *u* on item *j*
N(i;u)… set items rated by *u* similar to *i*

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)

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 of movie 1 by user 5

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	

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

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	

Compute similarity weights:

s_{13}=0.41, s_{16}=0.59

Item-Item CF (|N|=2)

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		**2.6**	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	

Predict by taking weighted average:

r_{51}=(0.41*2+0.59*3)/(0.41+0.59)=2.6

Before:

$$r_{ui} = \frac{\sum_{j \in N(i;u)} s_{ij} r_{uj}}{\sum_{j \in N(i;u)} s_{ij}}$$

CF: Common Practice

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

$$r_{ui} = b_{ui} + \frac{\sum_{j \in N(i;u)} s_{ij} \left(r_{uj} - b_{uj} \right)}{\sum_{j \in N(i;u)} s_{ij}}$$

baseline estimate for r_{ui}

$$b_{ui} = \mu + b_u + b_i$$

- μ = overall mean movie rating
- b_u = rating deviation of user *u*
- = *avg. rating of user u* − μ
- b_i = rating deviation of movie *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
  - 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

Finding Similar Vectors

- Common problem that comes up in many settings
- Given a large number *N* of vectors in some high-dimensional space (*M* dimensions), find pairs of vectors that have high similarity
  - e.g., user profiles, item profiles
- **We already know how to do this!**
  - Near-neighbor search in high dimensions (LSH)

Clustering Users and Items

- Hard to detect similarity among either items or users due to little information about user-item pairs.
- **Solution:** Cluster items and/or users
- Revise the utility matrix

The Netflix Prize

- **Training data**
  - 100 million ratings, 480,000 users, 17,770 movies
  - 6 years of data: 2000-2005
- **Test data**
  - Last few ratings of each user (2.8 million)
  - Evaluation criterion: Root Mean Square Error (**RMSE**)
  - Netflix Cinematch RMSE: 0.9514
- **Competition**
  - 2700+ teams
  - $1 million prize for 10% improvement on Cinematch

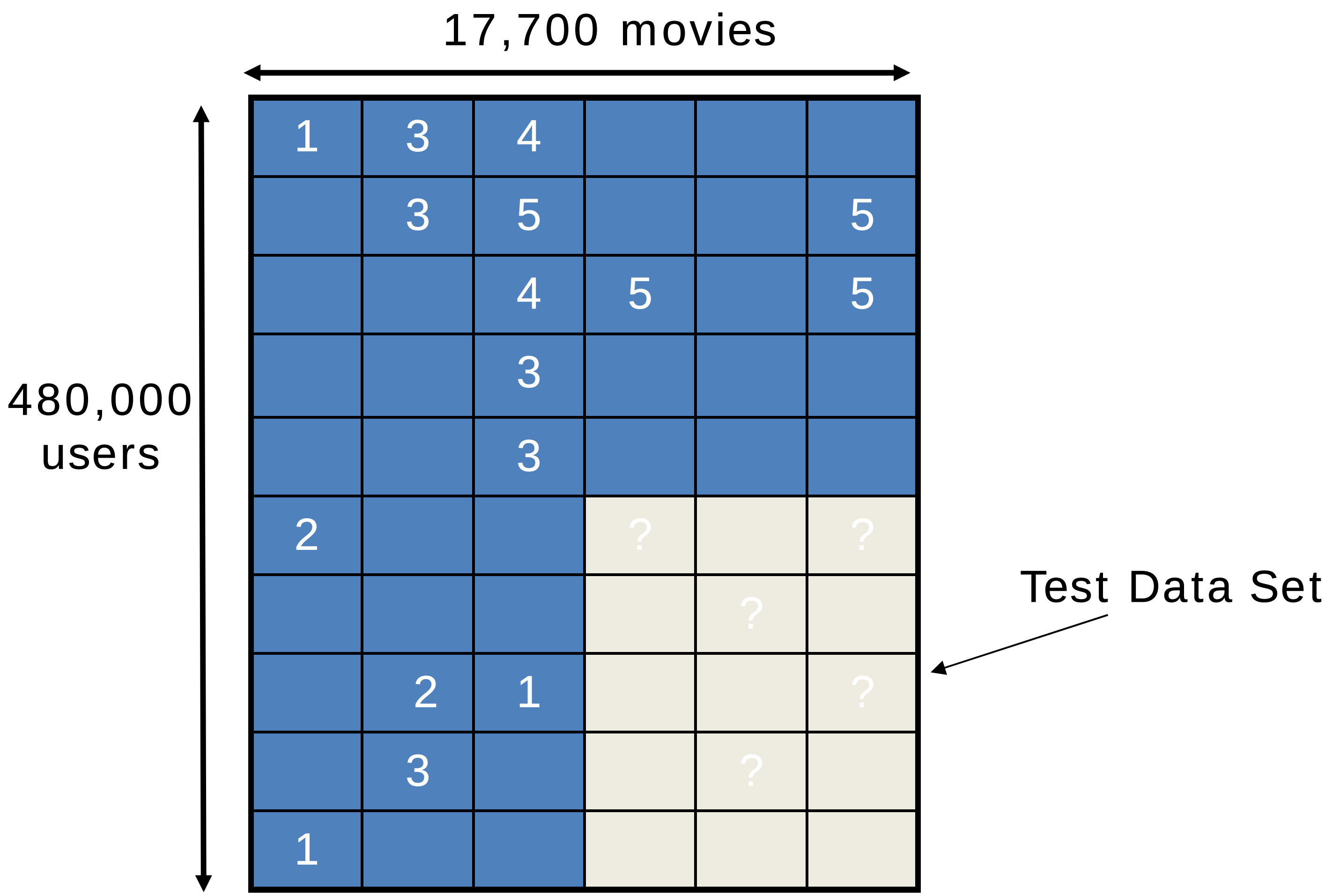
The Netflix Utility Matrix

17,700 movies

480,000 users

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

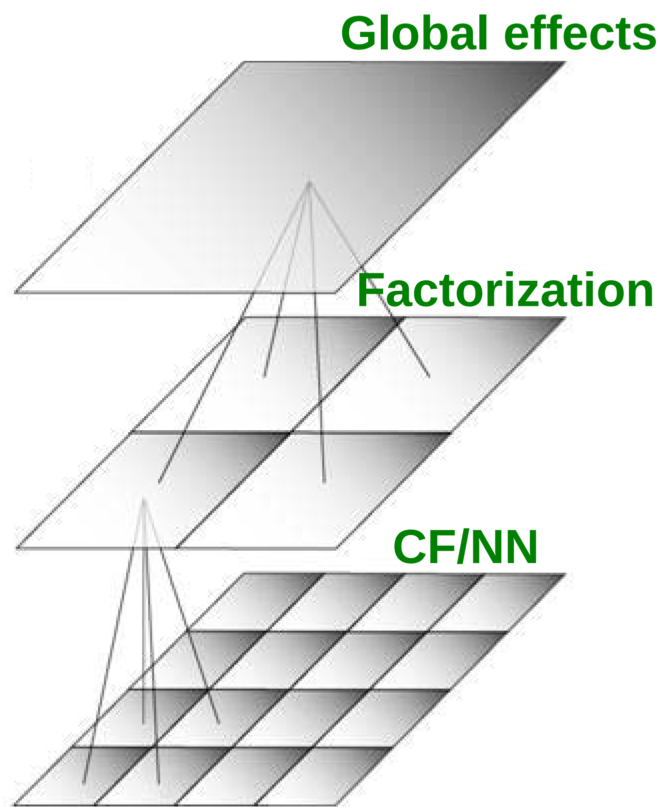
Utility Matrix: Evaluation



$$\textbf{SSE} = \sum_{(i,u)\in R}(r_{ui} - \hat{r}_{ui})^2$$

BellKor Recommender System

- **Basically the winner of the Netflix Challenge**
- Multi-scale modeling of the data: Combine top level, regional modeling of the data, with a refined, local view:
  - **Global:**
    - Overall deviations of users/movies
  - **Factorization:**
    - Addressing regional effects
  - **CF** (k-NN):
    - Extract local patterns



Evaluating Predictions

- **Compare predictions with known ratings**
  - **Root-mean-square error** (RMSE)
    - $\sqrt{\sum_{xi}(r_{xi} - r_{xi}^{*})^2}$ where $\boldsymbol{r}_{xi}$ is predicted, $\boldsymbol{r}_{xi}^{*}$ is the true rating of $\boldsymbol{x}$ on $\boldsymbol{i}$
  - **Precision at top 10**:
    - % of those in top 10
  - **Rank Correlation**:
    - Spearman's *correlation* between system's and user's complete rankings
- **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(N ·|C|)**
- **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