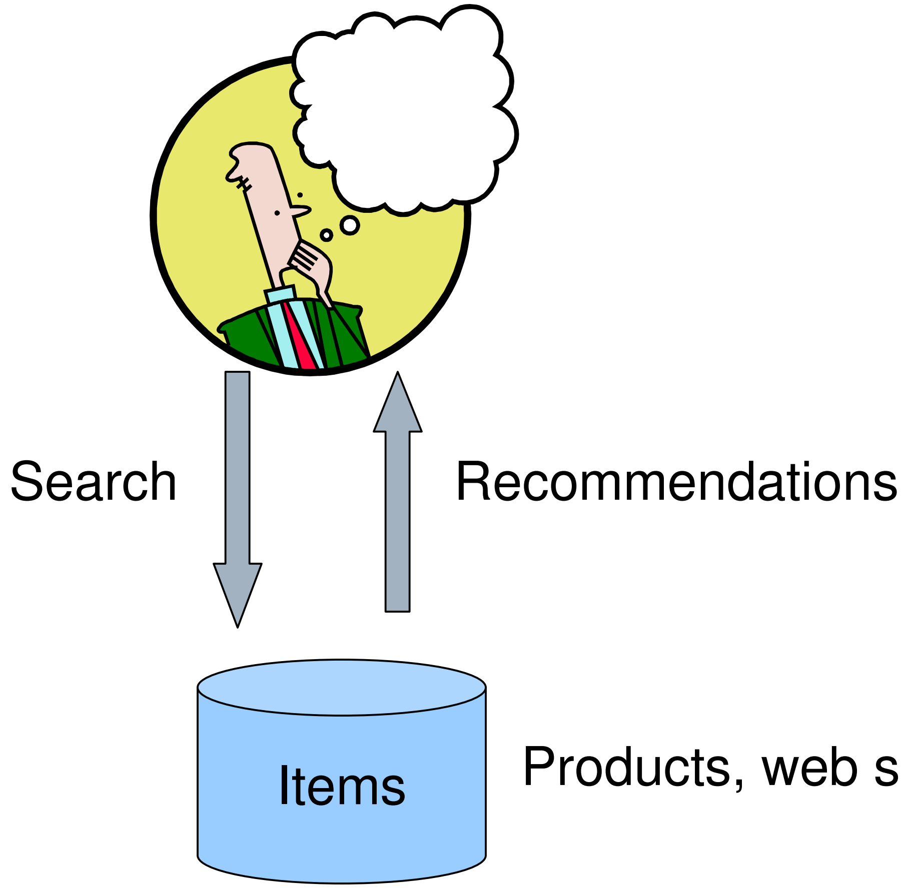
CS345
Data Mining

Recommendation Systems
Netflix Challenge

Anand Rajaraman, Jeffrey D. Ullman

Recommendations

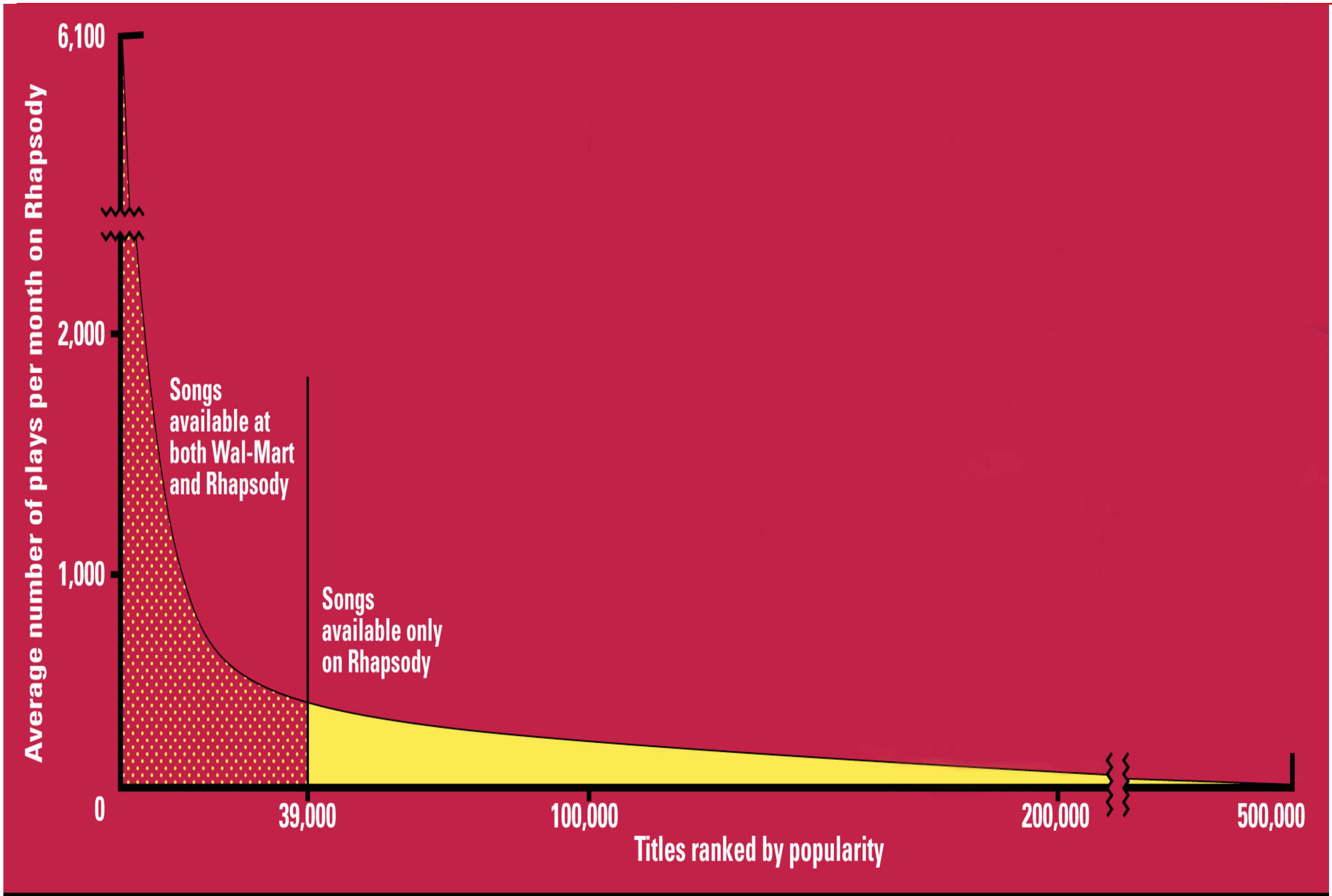


Products, web sites, blogs, news items, …

From scarcity to abundance

- Shelf space is a scarce commodity for traditional retailers
  - Also: TV networks, movie theaters,…
- The 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

The Long Tail



Source: Chris Anderson (2004)

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

Recommendation Types

- Editorial
- 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* £ *S* ! *R*
  - *R* = set of ratings
  - *R* is a totally ordered set
  - e.g., 0-5 stars, real number in [0,1]

Utility Matrix

	King Kong	LOTR	Matrix	Nacho Libre
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?

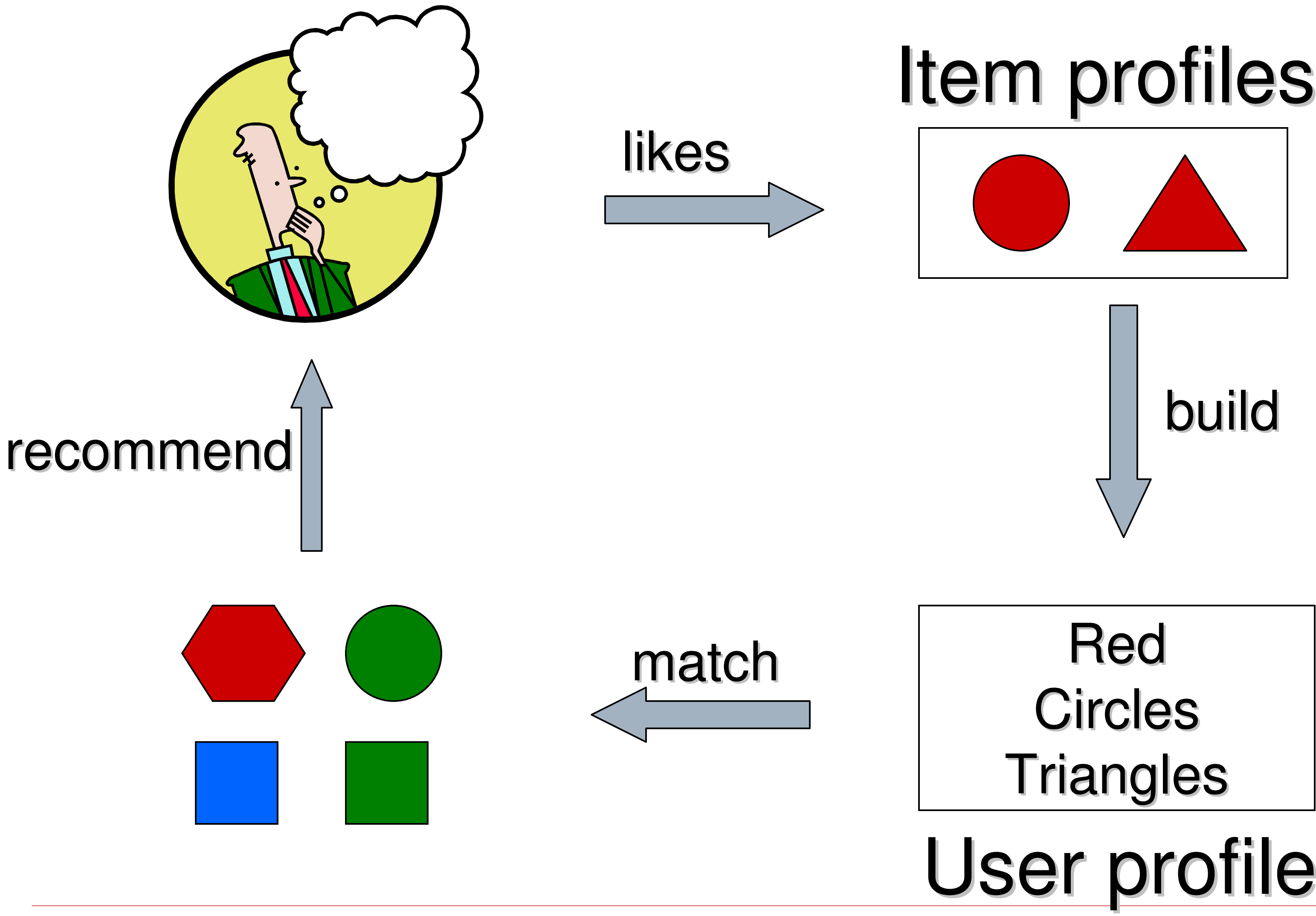
Extrapolating Utilities

- Key problem: matrix U is sparse
  - most people have not rated most items
- Three approaches
  - Content-based
  - Collaborative
  - Hybrid

Content-based recommendations

- Main idea: recommend items to customer C similar to previous items rated highly by C
- 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 of features
  - movies: author, title, actor, director,…
  - text: set of "important" words in document
- How to pick important words?
  - Usual heuristic is TF.IDF (Term Frequency times Inverse Doc Frequency)

TF.IDF

f_{ij} = frequency of term t_i in document d_j

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

n_i = number of docs that mention term i

N = total number of docs

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

TF.IDF score $w_{ij} = TF_{ij}$ £ 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 **c** and item profile **s**, estimate u(**c**,**s**) = cos(**c**,**s**) = **c**.**s**/(|**c**||**s**|)
  - Need efficient method to find items with high utility: later

Model-based approaches

- For each user, learn a classifier that classifies items into rating classes
  - liked by user and not liked by user
  - e.g., Bayesian, regression, SVM
- Apply classifier to each item to find recommendation candidates
- Problem: scalability
  - Won’t investigate further in this class

Limitations of content-based approach

- Finding the appropriate features
  - e.g., images, movies, music
- Overspecialization
  - Never recommends items outside user’s content profile
  - People might have multiple interests
- Recommendations for new users
  - How to build a profile?

Collaborative Filtering

- Consider user c
- Find set D of other users whose ratings are “similar” to c’s ratings
- Estimate user’s ratings based on ratings of users in D

Similar users

- Let r_x be the vector of user x's ratings
- Cosine similarity measure
  - $sim(x,y) = \cos(r_x, r_y)$

- Pearson correlation coefficient
  - S_{xy} = items rated by both users x and y

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

Rating predictions

- Let D be the set of *k* users most similar to *c* who have rated item *s*
- Possibilities for prediction function (item s):
  - $r_{cs} = 1/k \sum_{d \in D} r_{ds}$
  - $r_{cs} = (\sum_{d \in D} sim(c,d) \times r_{ds}) / (\sum_{d \in D} sim(c,d))$
  - Other options?
- Many tricks possible…

Complexity

- Expensive step is finding k most similar customers
  - O(|U|)
- Too expensive to do at runtime
  - Need to pre-compute
- Naïve precomputation takes time O(N|U|)
  - Simple trick gives some speedup
- Can use clustering, partitioning as alternatives, but quality degrades

Item-Item Collaborative Filtering

- So far: User-user collaborative filtering
- Another view
  - For item s, 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
- In practice, it has been observed that item-item often works better than user-user

Pros and cons of collaborative filtering

- Works for any kind of item
  - No feature selection needed
- New user problem
- New item problem
- Sparsity of rating matrix
  - Cluster-based smoothing?

Hybrid Methods

- Implement two separate recommenders and combine predictions
- Add content-based methods to collaborative filtering
  - item profiles for new item problem
  - demographics to deal with new user problem

Evaluating Predictions

- Compare predictions with known ratings
  - Root-mean-square error (RMSE)
- 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

Problems with Measures

- Narrow focus on accuracy sometimes misses the point
  - Prediction Diversity
  - Prediction Context
  - Order of predictions
- In practice, we care only to predict high ratings
  - RMSE might penalize a method that does well for high ratings and badly for others

Tip: Add data

- Leverage all the Netflix 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

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 cosine-similarity
  - e.g., user profiles, item profiles
- Perfect set-up for next topic!
  - Near-neighbor search in high dimensions