Recommender Systems
Example

- **Customer A**
  - Buys Metalica CD
  - Buys Megadeth CD

- **Customer B**
  - Does search on Metalica
  - Recommender system suggests Megadeth from data collected from customer A
Recommendations

Examples:

- Amazon.com
- Pandora
- StumbleUpon
- del.icio.us
- Netflix
- Movielens
  - helping you find the right movies
- Last.fm
- Google News
- YouTube
- Xbox Live

Search

Recommendations

Items

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:
    - [http://www.wired.com/wired/archive/12.10/tail.html](http://www.wired.com/wired/archive/12.10/tail.html)
The Long Tail

Source: Chris Anderson (2004)
Physical vs. Online

The Bit Player Advantage

Beyond bricks and mortar there are two main retail models – one that gets halfway down the Long Tail and another that goes all the way. The first is the familiar hybrid model of Amazon and Netflix, companies that sell physical goods online. Digital catalogs allow them to offer unlimited selection along with search, reviews, and recommendations, while the cost savings of massive warehouses and no walk-in customers greatly expands the number of products they can sell profitably. Pushing this even further are pure digital services, such as iTunes, which offer the additional savings of delivering their digital goods online at virtually no marginal cost. Since an extra database entry and a few megabytes of storage on a server cost effectively nothing, these retailers have no economic reason not to carry everything available.

“IF YOU LIKE BRITNEY, YOU’LL LOVE …”

Just as lower prices can entice consumers down the Long Tail, recommendation engines drive them to obscure content they might not find otherwise.

Read [http://www.wired.com/wired/archive/12.10/tail.html](http://www.wired.com/wired/archive/12.10/tail.html) to learn more!
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 \( u: C \times S \rightarrow R \)
  - \( R = \) set of ratings
  - \( R \) is a totally ordered set
  - e.g., 0-5 stars, real number in [0,1]
## Utility Matrix

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alice</strong></td>
<td>1</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Bob</strong></td>
<td>0.5</td>
<td></td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td><strong>Carol</strong></td>
<td>0.2</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td><strong>David</strong></td>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
</tr>
</tbody>
</table>
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

- **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

likes

build

match

recommend

Red Circles

Triangles

User profile
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**

\[ f_{ij} = \text{frequency of term } t_i \text{ in document } d_j \]

\[ TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}} \]

\[ n_i = \text{number of docs that mention term } i \]

\[ N = \text{total number of docs} \]

\[ IDF_i = \log \frac{N}{n_i} \]

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 \( c \) and item profile \( s \), estimate
    \[ u(c,s) = \cos(c,s) = \frac{c.s}{|c| |s|} \]
  - Need efficient method to find items with high utility: later
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
  - 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 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
  - $\text{sim}(x, y) = \cos(r_x, r_y)$

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

\[
\text{sim}(x, y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)(r_{ys} - \bar{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)^2(r_{ys} - \bar{r}_y)^2}}
\]
Let $D$ be the set of $k$ users most similar to $c$ who have rated item $s$

Possibilities for prediction function (item $s$):

- $r_{cs} = \frac{1}{k} \sum_{d \in D} r_{ds}$
- $r_{cs} = \frac{\sum_{d \in D} \text{sim}(c,d) \ r_{ds}}{\sum_{d \in D} \text{sim}(c,d)}$

Other options?

Many tricks possible...
Expensive step is finding k most similar customers
- $O(|U|)$

Too expensive to do at runtime
- Could pre-compute

Naïve precomputation takes time $O(N|U|)$
- Stay tuned for how to do it faster!

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
**Example: Item-Item based**

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Bob</td>
<td>0.5</td>
<td>0.3</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td>0.2</td>
<td>1</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>David</td>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
</tr>
</tbody>
</table>

- What do we recommend for Avatar?
  - $\cos(\text{Avatar, Matrix}) = 0.38$
  - $\cos(\text{Avatar, Lotr}) = 0.0$
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 tastes.
  - 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
Evaluation

<table>
<thead>
<tr>
<th>1</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
## Evaluation

**Test Data Set** (most recent ratings)

<table>
<thead>
<tr>
<th>users</th>
<th>movies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 4</td>
</tr>
<tr>
<td>3 5 5</td>
<td>4 5 5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>? ?</td>
</tr>
<tr>
<td>2 1</td>
<td>?</td>
</tr>
<tr>
<td>3</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Evaluating Predictions

- Compare predictions with known ratings
  - Root-mean-square error (RMSE)
  - Precision at top 10: % of those in top10
  - Rating of top 10: Average rating assigned to top 10
  - Rank Correlation: Spearman’s, $r_s$, 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
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
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)
  - Dimensionality reduction
The bubbles above represent products sized by sales volume. Products close to each other are recommended to each other.