15. Recommender Systems

These notes are based, in part, on notes by Dr. Raymond J. Mooney at the University of Texas at Austin.
Recommender Systems

• Systems for recommending items (e.g. books, movies, CD’s, web pages, newsgroup messages) to users based on examples of their preferences.

• Many websites provide recommendations (e.g. Amazon, NetFlix, Pandora).

• Recommenders have been shown to substantially increase sales at on-line stores.

• There are two basic approaches to recommending:
  – Collaborative Filtering (a.k.a. social filtering)
  – Content-based
Collaborative Filtering

“Social Learning”
- idea is to give recommendations to a user based on the “ratings” of objects by other users
- usually assumes that features in the data are similar objects (e.g., Web pages, music, movies, etc.)
- usually requires “explicit” ratings of objects by users based on a rating scale
- there have been some attempts to obtain ratings implicitly based on user behavior (mixed results; problem is that implicit ratings are often binary)

<table>
<thead>
<tr>
<th></th>
<th>Star Wars</th>
<th>Jurassic Park</th>
<th>Terminator 2</th>
<th>Indep. Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sally</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Bob</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Chris</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>Lynn</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Karen</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>?</td>
</tr>
</tbody>
</table>

Will Karen like “Independence Day?”
Collaborative Recommender Systems
Collaborative Recommender Systems
Collaborative Recommender Systems

Get Recommendations (204)  Rate Movies  Movies You’ve Rated (104)

ALL RECOMMENDATIONS
Get more Recommendations by rating more movies.

**Gladiator: Extended Edition**

Fans of Gladiator's original theatrical release will appreciate this extended version of the epic Ridley Scott film, packed with 17 extra minutes of action footage and gripping dialogue. Featuring a strong supporting cast and an Oscar-winning performance from actor Russell Crowe as the dauntless Roman general Maximus, this big-budget Best Picture winner became an instant classic -- and helped elevate its leading man to icon status.

Starring: Russell Crowe, Joaquin Phoenix
Director: Ridley Scott

**Blade Runner: The Director's Cut**

In the smog-choked dystopian Los Angeles of 2019, blade runner Rick Deckard (Harrison Ford) is called out of retirement to snuff a quartet of “replicants” -- androids consigned to slave labor on remote planets. They've escaped to Earth seeking their creator and a way to extend their short life spans. Director Ridley Scott's reedited version comes with a different ending and the omission of Ford's narration, giving the film a different tone.

Starring: Harrison Ford, Rutger Hauer
Director: Ridley Scott

**The Shawshank Redemption: Special Edition**

Upstanding banker Andy Dufresne (Tim Robbins) is framed for a double murder in the 1940s and begins a life sentence at the Shawshank prison, where he's befriended by an older inmate named Red (Morgan Freeman). During his long stretch in prison, Dufresne comes to be admired by the other inmates for his upstanding moral code and unquenchable sense of hope. Co-stars Gil Bellows and Bob Gunton (who's memorable as the amoral prison warden).
Collaborative Filtering: Nearest-Neighbor Strategy

• Basic Idea:
  ‣ find other users that are most similar preferences or tastes to the target user
  ‣ Need a metric to compute similarities among users (usually based on their ratings of items)

• Pearson Correlation
  ‣ weight by degree of correlation between user U and user J

\[
 r_{UJ} = \frac{\sum (U - \overline{U})(J - \overline{J})}{\sqrt{\sum (U - \overline{U})^2 \cdot \sum (J - \overline{J})^2}}
\]

• 1 means very similar, 0 means no correlation, -1 means dissimilar

Average rating of user J on all items.
Collaborative Filtering: Making Predictions

- When generating predictions from the nearest neighbors, neighbors can be weighted based on their distance to the target user.
- To generate predictions for a target user $a$ on an item $i$:

$$ p_{a,i} = \bar{r}_a + \sum_{u=1}^{k} \frac{(r_{u,i} - \bar{r}_u) \times \text{sim}(a,u)}{\sum_{u=1}^{k} \text{sim}(a,u)} $$

- $\bar{r}_a$ = mean rating for user $a$
- $u_1, ..., u_k$ are the $k$-nearest-neighbors to $a$
- $r_{u,i}$ = rating of user $u$ on item $I$
- $\text{sim}(a,u)$ = Pearson correlation between $a$ and $u$

- This is a weighted average of deviations from the neighbors’ mean ratings (and closer neighbors count more)
Distance or Similarity Measures

- **Pearson Correlation**
  - Works well in case of user ratings (where there is at least a range of 1-5)
  - Not always possible (in some situations we may only have implicit binary values, e.g., whether a user did or did not select a document)
  - Alternatively, a variety of distance or similarity measures can be used

- **Common Distance Measures:**

  - **Manhattan distance:**
    \[
    \text{dist}(X,Y) = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_n - y_n|
    \]

  - **Euclidean distance:**
    \[
    \text{dist}(X,Y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}
    \]

  - **Cosine similarity:**
    \[
    \text{dist}(X,Y) = 1 - \text{sim}(X,Y)
    \]
    \[
    \text{sim}(X,Y) = \frac{\sum_i (x_i \times y_i)}{\sqrt{\sum_i x_i^2 \times \sum_i y_i^2}}
    \]
**Example Collaborative System**

<table>
<thead>
<tr>
<th></th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item 4</th>
<th>Item 5</th>
<th>Item 6</th>
<th>Correlation with Alice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>User 1</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>-1.00</td>
<td></td>
</tr>
<tr>
<td>User 2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>User 3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>.90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User 4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
<td>0.19</td>
<td></td>
</tr>
<tr>
<td>User 5</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User 6</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User 7</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Using k-nearest neighbor with k = 1*
Item-based Collaborative Filtering

- Find similarities among the items based on ratings across users
  - Often measured based on a variation of Cosine measure
- Prediction of item I for user \( a \) is based on the past ratings of user \( a \) on items similar to \( i \).

<table>
<thead>
<tr>
<th></th>
<th>Star Wars</th>
<th>Jurassic Park</th>
<th>Terminator 2</th>
<th>Indep. Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sally</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Bob</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Chris</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>Lynn</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Karen</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>?</td>
</tr>
</tbody>
</table>

Suppose: \( \text{sim(Star Wars, Indep. Day)} > \text{sim(Jur. Park, Indep. Day)} > \text{sim(Termin., Indep. Day)} \)

Predicted rating for Karen on Indep. Day will be 7, because she rated Star Wars 7
  - That is if we only use the most similar item
  - Otherwise, we can use the k-most similar items and again use a weighted average
# Item-Based Collaborative Filtering

[Table showing user-item interactions with item similarity scores and predicted ratings]

## Prediction

Cosine Similarity to the target item
Collaborative Filtering: Pros & Cons

• **Advantages**
  
  - Ignores the content, only looks at who judges things similarly
    
    • If Pam liked the paper, I’ll like the paper
    
    • If you liked Star Wars, you’ll like Independence Day
    
    • Rating based on ratings of similar people
  
  - Works well on data relating to “taste”
    
    • Something that people are good at predicting about each other too
    
    • can be combined with meta information about objects to increase accuracy

• **Disadvantages**
  
  - early ratings by users can bias ratings of future users
  
  - small number of users relative to number of items may result in poor performance
  
  - scalability problems: as number of users increase, nearest neighbor calculations become computationally intensive
  
  - because of the (dynamic) nature of the application, it is difficult to select only a portion instances as the training set.
Content-based recommendation

• Collaborative filtering does **NOT** require any information about the items,
  • However, it might be reasonable to exploit such information
  • E.g. recommend fantasy novels to people who liked fantasy novels in the past

• **What do we need:**
  • Some information about the available items such as the genre ("content")
  • Some sort of *user profile* describing what the user likes (the preferences)

• **The task:**
  • Learn user preferences
  • Locate/recommend items that are "similar" to the user preferences
Content-Based Recommenders

- Predictions for unseen (target) items are computed based on their similarity (in terms of content) to items in the user profile.
- E.g., user profile $P_u$ contains

recommend highly:  
and recommend “mildly”:
Content-based recommendation

• Basic approach

  ▶ Represent items as vectors over features
  ▶ User profiles are also represented as aggregate feature vectors
    • Based on items in the user profile (e.g., items liked, purchased, viewed, clicked on, etc.)
  ▶ Compute the similarity of an unseen item with the user profile based on the keyword overlap (e.g. using the Dice coefficient)

\[
\text{sim}(b_i, b_j) = \frac{2 \times |\text{keywords}(b_i) \cap \text{keywords}(b_j)|}{|\text{keywords}(b_i)| + |\text{keywords}(b_j)|}
\]

  ▶ Other similarity measures such as Cosine can also be used
  ▶ Recommend items most similar to the user profile
Content-Based Recommender Systems
Content-Based Recommenders: Personalized Search

• How can the search engine determine the “user’s context”?

Query: “Madonna and Child”

• Need to “learn” the user profile:
  ▶ User is an art historian?
  ▶ User is a pop music fan?
Content-Based Recommenders

- Music recommendations
- Play list generation

Example: Pandora
Advantages of 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.
Disadvantages of Content-Based Method

• Requires content that can be encoded as meaningful features.
• Users’ tastes must be represented as a learnable function of these content features.
• Unable to exploit quality judgments of other users.
  – Unless these are somehow included in the content features.
Social / Collaborative Tags
Example: Tags describe the Resource

Tags can describe
- The resource (genre, actors, etc)
- Organizational (toRead)
- Subjective (awesome)
- Ownership (abc)
- etc
Tag Recommendation
These systems are “collaborative.”

- Recommendation / Analytics based on the “wisdom of crowds.”

Rai Aren's profile
cos-author

Location: Canada
Web Page: www.secretofthesands.com
In My Own Words: 
RAI AREN

Rai loves the stories of Lord of the Rings, Star Wars, Star Trek, Indiana Jones (her first kitty cat is named Indiana, Indy for short), and The Matrix (take the red pill!), to name a few. She loves getting lost in these enchanting worlds and studying their underlying philosophies. Ancient Egypt has held a particular fascination for her since childhood.

Rai feels that novels have the abi...
Read more

See all 1,832 tagged items
Social Recommendation

- A form of collaborative filtering using social network data
  - Users profiles represented as sets of links to other nodes (users or items) in the network
  - Prediction problem: infer a currently non-existent link in the network
Example: Using Tags for Recommendation

Last.fm recommendations

- Recommendations:
  - Primarily Collaborative Filtering
  - Item-Item (artist recommendations)
  - User-User (Neighbors)
  - Could use: tags, audio, metadata

- Evaluating (rel. feedback)
  - Tracking Love/Ban behavior
Learning interface agents

- Add agents to the user interface and delegate tasks to them
- **Use machine learning to improve performance**
  - learn user behavior, preferences
- **Useful when:**
  - 1) past behavior is a useful predictor of the future behavior
  - 2) wide variety of behaviors amongst users
- **Examples:**
  - mail clerk: sort incoming messages in right mailboxes
  - calendar manager: automatically schedule meeting times?
  - Personal news agents
  - portfolio manager agents
- **Advantages:**
  - less work for user and application writer
  - adaptive behavior
  - user and agent build trust relationship gradually
Letizia: Autonomous Interface Agent
(Lieberman 96)

- Recommends web pages during browsing based on user profile
- Learns user profile using simple heuristics
- Passive observation, recommend on request
- Provides *relative* ordering of link interestingness
- Assumes recommendations “near” current page are more valuable than others
Letizia: Autonomous Interface Agent

- Infers user preferences from behavior
- Interesting pages
  - record in hot list (save as a file)
  - follow several links from pages
  - returning several times to a document
- Not Interesting
  - spend a short time on document
  - return to previous document without following links
  - passing over a link to document (selecting links above and below document)
- Why is this useful
  - tracks and learns user behavior, provides user “context” to the application (browsing)
  - completely passive: no work for the user
  - useful when user doesn’t know where to go
  - no modifications to application: Letizia interposes between the Web and browser
Consequences of passiveness

• Weak heuristics
  ‣ example: click through multiple uninteresting pages en route to interestingness
  ‣ example: user browses to uninteresting page, then goes for a coffee
  ‣ example: hierarchies tend to get more hits near root

• Cold start

• No ability to fine tune profile or express interest without visiting “appropriate” pages

• Some possible alternative/extensions to internally maintained profiles:
  ‣ expose to the user (e.g. fine tune profile) ?
  ‣ expose to other users/agents (e.g. collaborative filtering)?
  ‣ expose to web server (e.g. cnn.com custom news)?
ARCH: Adaptive Agent for Retrieval Based on Concept Hierarchies
(Mobasher, Sieg, Burke 2003-2007)

• ARCH supports users in formulating effective search queries starting with users’ poorly designed keyword queries
• Essence of the system is to incorporate domain-specific concept hierarchies with interactive query formulation
• Query enhancement in ARCH uses two mutually-supporting techniques:
  ‣ Semantic – using a concept hierarchy to interactively disambiguate and expand queries
  ‣ Behavioral – observing user’s past browsing behavior for user profiling and automatic query enhancement
Overview of ARCH

• The system consists of an offline and an online component

• Offline component:
  ▹ Handles the learning of the concept hierarchy
  ▹ Handles the learning of the user profiles

• Online component:
  ▹ Displays the concept hierarchy to the user
  ▹ Allows the user to select/deselect nodes
  ▹ Generates the enhanced query based on the user’s interaction with the concept hierarchy
Offline Component - Learning the Concept Hierarchy

- Maintain aggregate representation of the concept hierarchy
  - pre-compute the term vectors for each node in the hierarchy
  - Concept classification hierarchy - Yahoo

![Diagram of concept hierarchy]

<table>
<thead>
<tr>
<th>Language</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>0.452</td>
</tr>
<tr>
<td>C#</td>
<td>0.364</td>
</tr>
<tr>
<td>Java</td>
<td>0.297</td>
</tr>
<tr>
<td>Library</td>
<td>0.256</td>
</tr>
<tr>
<td>Code</td>
<td>0.364</td>
</tr>
<tr>
<td>Program</td>
<td>0.38</td>
</tr>
<tr>
<td>Example</td>
<td>0.298</td>
</tr>
<tr>
<td>Document</td>
<td>0.266</td>
</tr>
<tr>
<td>Compile</td>
<td>0.283</td>
</tr>
<tr>
<td>Implement</td>
<td>0.315</td>
</tr>
<tr>
<td>User</td>
<td>0.251</td>
</tr>
<tr>
<td>Interpret</td>
<td>0.316</td>
</tr>
</tbody>
</table>

Intelligent Information Retrieval
Aggregate Representation of Nodes in the Hierarchy

- A node is represented as a weighted term vector:
  - centroid of all documents and subcategories indexed under the node

\[
T_n = \left( \frac{\left( \sum_{d \in D_n} T_d \right)}{|D_n| + \sum_{s \in S_n} T_s} \right) / (|S_n| + 1)
\]

\(n\) = node in the concept hierarchy
\(D_n\) = collection of individual documents
\(S_n\) = subcategories under \(n\)
\(T_d\) = weighted term vector for document \(d\) indexed under node \(n\)
\(T_s\) = the term vector for subcategory \(s\) of node \(n\)
Example from Yahoo Hierarchy

Term Vector for "Genres:"

- music: 1.000
- blue: 0.15
- new: 0.14
- artist: 0.13
- jazz: 0.12
- review: 0.12
- band: 0.11
- polka: 0.10
- festiv: 0.10
- celtic: 0.10
- freestyl: 0.10
Online Component – User Interaction with Hierarchy

• The initial user query is mapped to the relevant portions of hierarchy
  ‣ user enters a keyword query
  ‣ system matches the term vectors representing each node in the hierarchy with the keyword query
  ‣ nodes which exceed a similarity threshold are displayed to the user, along with other adjacent nodes.

• Semi-automatic derivation of user context
  ‣ ambiguous keyword might cause the system to display several different portions of the hierarchy
  ‣ user selects categories which are relevant to the intended query, and deselects categories which are not
Generating the Enhanced Query

• Based on an adaptation of Rocchio's method for relevance feedback
  ‣ Using the selected and deselected nodes, the system produces a refined query $Q_2$:

$$Q_2 = \alpha \cdot Q_1 + \beta \cdot \sum T_{sel} - \gamma \cdot \sum T_{desel}$$

• each $T_{sel}$ is a term vector for one of the nodes selected by the user,
• each $T_{desel}$ is a term vector for one of the deselected nodes
• factors $\alpha$, $\beta$, and $\gamma$ are tuning parameters representing the relative weights associated with the initial query, positive feedback, and negative feedback, respectively such that $\alpha + \beta - \gamma = 1$. 
An Example

Initial Query  "music, jazz"

Selected Categories
"Music", "jazz", "Dixieland"

Deselected Category
"Blues"

Portion of the resulting term vector:

music: 1.00, jazz: 0.44, dixieland: 0.20, tradition: 0.11, band: 0.10, inform: 0.10, new: 0.07, artist: 0.06
Another Example – ARCH Interface

• Initial query = *python*
• Intent for search = *python* as a *snake*
• User selects *Pythons* under *Reptiles*
• User deselects *Python* under *Programming and Development* and *Monty Python* under *Entertainment*

• Enhanced query:

<table>
<thead>
<tr>
<th>Term</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>burmes</td>
<td>1</td>
</tr>
<tr>
<td>python</td>
<td>0.937</td>
</tr>
<tr>
<td>snake</td>
<td>0.733</td>
</tr>
<tr>
<td>reptil</td>
<td>0.499</td>
</tr>
<tr>
<td>infect</td>
<td>0.494</td>
</tr>
<tr>
<td>constrictor</td>
<td>0.42</td>
</tr>
<tr>
<td>vet</td>
<td>0.472</td>
</tr>
<tr>
<td>lizard</td>
<td>0.472</td>
</tr>
<tr>
<td>pet</td>
<td>0.41</td>
</tr>
<tr>
<td>egg</td>
<td>0.43</td>
</tr>
</tbody>
</table>
Generation of User Profiles

• **Profile Generation Component of ARCH**
  - passively observe user’s browsing behavior
  - use heuristics to determine which pages user finds “interesting”
    - time spent on the page (or similar pages)
    - frequency of visit to the page or the site
    - other factors, e.g., bookmarking a page, etc.
  - implemented as a client-side proxy server

• **Clustering of “Interesting” Documents**
  - ARCH extracts feature vectors for each profile document
  - documents are clustered into semantically related categories
    - we use a clustering algorithm that supports overlapping categories to capture relationships across clusters
    - algorithms: overlapping version of k-means; hypergraph partitioning
  - profiles are the significant features in the centroid of each cluster
User Profiles & Information Context

• Can user profiles replace the need for user interaction?
  ‣ Instead of explicit user feedback, the user profiles are used for the selection and deselection of concepts
  ‣ Each individual profile is compared to the original user query for similarity
  ‣ Those profiles which satisfy a similarity threshold are then compared to the matching nodes in the concept hierarchy
    • matching nodes include those that exceeded a similarity threshold when compared to the user’s original keyword query.
  ‣ The node with the highest similarity score is used for automatic selection; nodes with relatively low similarity scores are used for automatic deselection
Results Based on User Profiles

Simple vs. Enhanced Query Search

- Simple Query Single Keyword
- Simple Query Two Keywords
- Enhanced Query with User Profiles

Precision vs. Threshold (%)

Recall vs. Threshold (%)

Intelligent Information Retrieval