COEN 169

Recommendation Systems II

Yi Fang

Department of Computer Engineering

Santa Clara University

User-based Collaborative Filtering

“Similar users rate similarly!”

Finding Similar Users

- Cosine similarity
- Pearson correlation
- Euclidean distance score
- …

Euclidean Distance Score

- The straight-line distance between two points in a multidimensional space, which is the kind of distance you measure with a ruler.

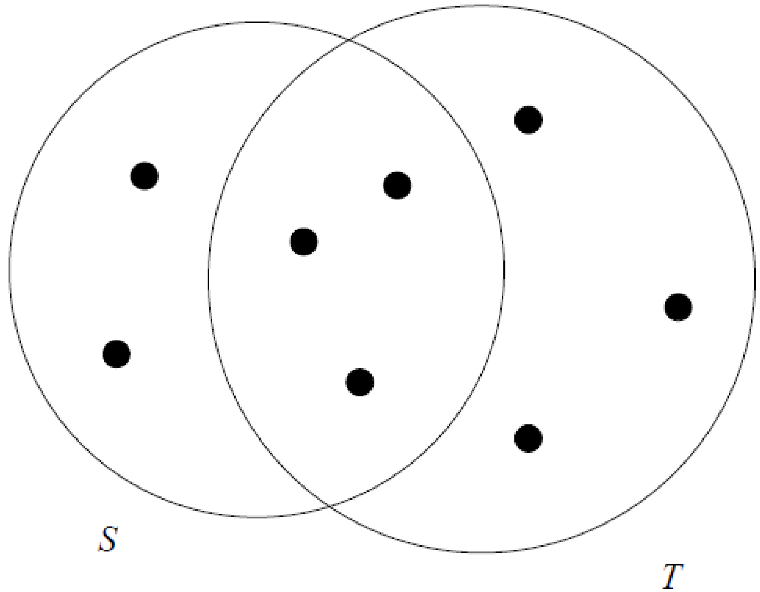
$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

- Given the Euclidean distance *d* between two points, how can we define their similarity score?
- Can be defined as $1/(d+1)$.

Jaccard Similarity

- For two sets A and B, Jaccard Similarity is defined as

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$



$sim(A,B) = 3/8$

- Often used for binary ratings

User-based Collaborative Filtering

- Step 1: Look for users who share the same rating patterns with the active user
  - e.g., using the k-Nearest Neighbours algorithm

- Step 2: Use the ratings from those like-minded users to calculate a prediction for the active user.

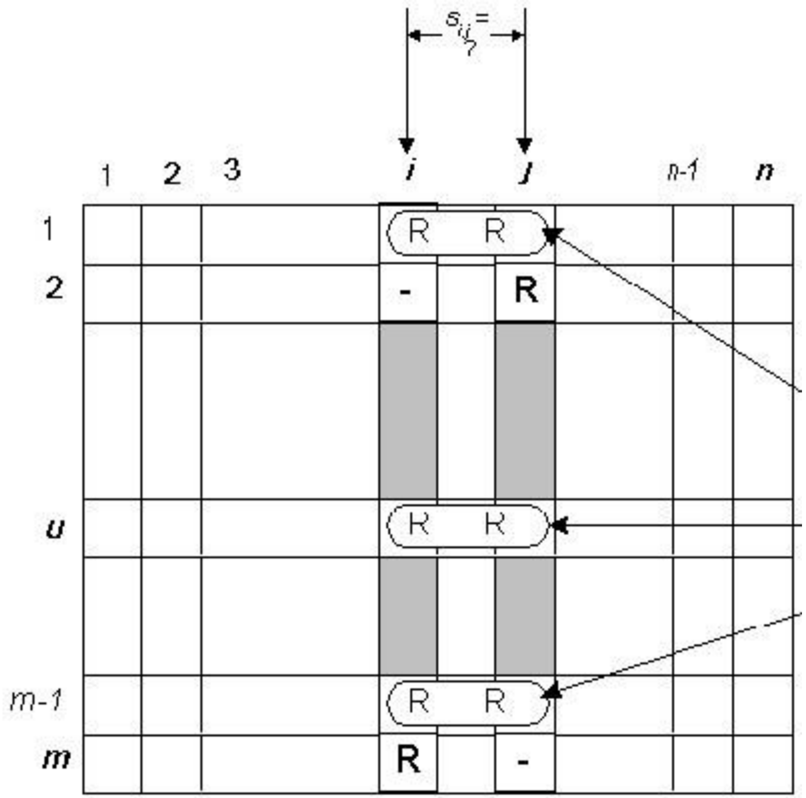
Item-based Collaborative Filtering

"Similar items are rated similarly!"

Item-based Collaborative Filtering

- Rather than matching the active user to similar customers, finding items that get similar ratings

Finding Similar Items



Computed by looking into co-rated items only. These co-rated pairs are obtained from different users.

Finding Similar Items

Just switch users and items in the previous slides!

Item-based CF: Cosine-based Similarity

$$sim(i_1, i_2) = \cos\vartheta_{i_1, i_2}$$

$$= \frac{\sum_{j=1}^{m} r_{u_j, i_1} \times r_{u_j, i_2}}{\sqrt{\sum_{j=1}^{m} r_{u_j, i_1}^2} \times \sqrt{\sum_{i=1}^{m} r_{u_j, i_1}^2}}$$

Item-based CF: Pearson Correlation Similarity

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_i)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_j)^2}}$$

Item-based CF: Adjusted Cosine Similarity

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R_u})(R_{u,j} - \bar{R_u})}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R_u})^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R_u})^2}}.$$

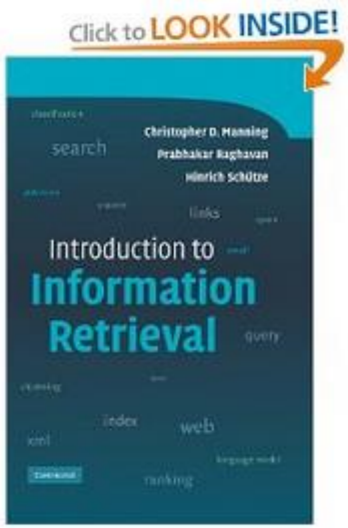
Item-based CF: Rating Prediction

- Predict a rating, $p_{a,i}$, for each item i and active user a by

$$p_{a,i} = \frac{\sum_{j=1}^{k} w_{i,j} r_{a,j}}{\sum_{u=1}^{k} | w_{i,j} |}$$

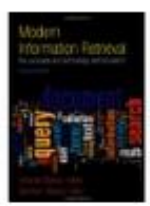
Amazon’s book recommendation

“Users who bought this book, also bought that book”



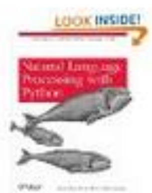
Customers Who Bought This Item Also Bought

Speech and Language Processing (2nd Edition)
Daniel Jurafsky
(32)
Hardcover
$112.47

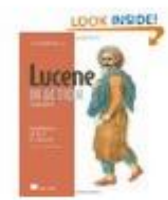


Modern Information Retrieval: The Concepts ...
› Ricardo Baeza-Yates
(1)
Paperback
$55.68

Foundations of Statistical Natural Language ...
› Christopher D. Manning
(14)
Hardcover
$56.84



Natural Language Processing with Python
› Steven Bird
(16)
Paperback
$37.59



Lucene in Action, Second Edition: Covers Apache ...
› Michael McCandless
(30)
Paperback
$31.36

User-based vs Item-based

- Efficiency
  - The latter is usually more efficient than the former
  - More users than items
- Effectiveness
  - A user may have multiple interests. Item similarity is more stable
  - You don't get very much diversity or surprise in item based recommendations, so recommendations tend to be kind of "obvious" and boring

Programming Assignment

- Choose ANY programming language
- Feel free to submit your results (up to 30 times)
- The basic algorithm is straightforward, but lots of room to tweak the performance
- Neighborhood approach can generate very good results if well tuned
- For Q3, grading will be based on performance OR novelty
- Write the report to summarize your results

Basic algorithm

1. Read the data from the files
2. Given a test user, compute similarities between users (user-based CF) or items (items-based CF)
3. Find the top *k* similar users or items in the training data
4. Use the *k* similar users (or items)'s ratings for predicting the ratings of the test user

Cold start problem

- New items or users have no historical ratings

Content based recommendations

- Recommend items based on content

  - Text documents are recommended based on a comparison between their content and a user profile
- Examples:

- LinkedIn’s job recommendations

IR techniques apply

- Vector space model
- Treat a user as a query, and treat an item as a document
- Item or user is represented by a vector of features
  - Text: TF-IDF scores of the words in the content description
- Given user vector *u* and item vector *i*, compute the cosine similarity as the recommendation score

Pros: Content-based approach

- No need for data on other users
- Able to recommend new and unpopular items
  - No cold-start or sparsity problems
- Interpretability: can provide explanations of recommended items by listing content-features that caused an item to be recommended

Cons: Content-based approach

- Never recommends items outside user's content profile
- In many applications, user profiles are not available
- Unable to exploit quality judgments of other users