# Recommendation (Personalization)

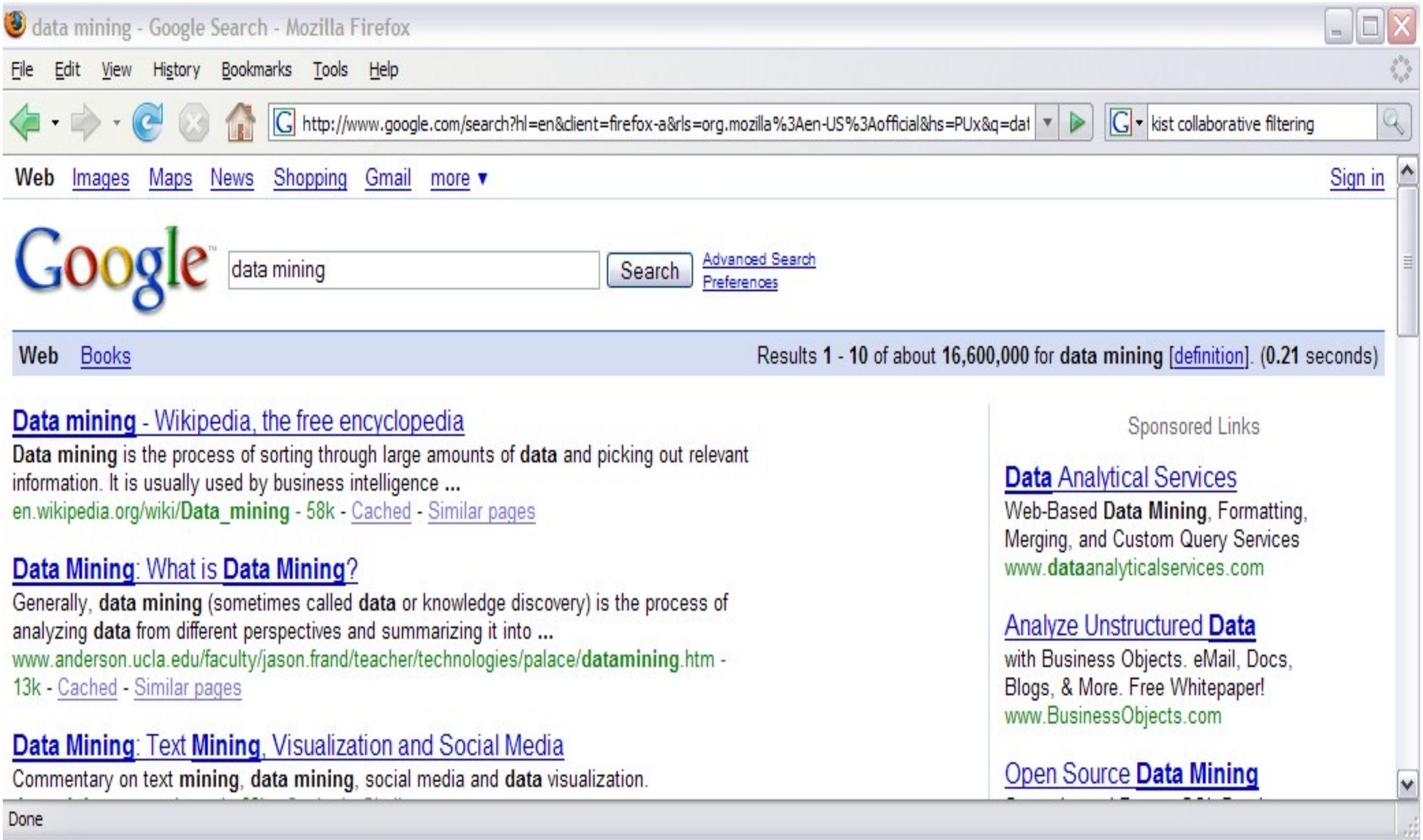
Man-Kwan Shan
Dept. of Computer Science
National Cheng-Chi Univ.

# Overview

# Recommendation

- Recommendation (Personalization)
  - recommendation made by system based on user preference
  - approaches
    - Content-based filtering: based on user preference in the past
    - Collaborative filtering: based on people with similar preference
    - Hybrid
- Example
  - personalized newspaper
  - personalized web
  - TV, Movies, Books, Music recommendations
  - personalized AD(Amazon)
  - personalized shopping mall(IBM IRA)

# Recommendation (cont.)



# Recommendation (cont.)

# Recommendation (cont.)

- Recommendation

$$\forall c \in C, s'_c = \arg\max_{s \in S} u(c, s)$$

  - $C$ : the set of all users
  - $S$ : the set of all possible items
  - $u$ : utility function that measures the usefulness of item $s$ to user $c$

  i.e. $u$: $C$ x $S \rightarrow R$,
  ($R$: nonnegative integers, a totally ordered set)
- In most cases
  - $c$: user profile, user characteristics
  - $s$: item characteristics
  - $u$: specified by user or computed by the application

# Recommendation (cont.)

- Central problem of recommendation
  - utility is usually not defined on the whole C x S space but only only on some subset of it
  - utility is initially defined only on the items previously rated by the users
  - should be able to estimate (predicate) the ratings of the non-rated movie/user combinations

# Content-based Recommendation

# Content-based Recommendation

- Content-based
  - recommend items similar to the ones the user preferred in the past
  - the utility $u(c, s)$ of item $s$ for user $c$ is estimated based on the utilities $u(c, s_i)$ assigned by user $c$ to items $s_i \in C$
  - try to understand the commonalities among the items user has rated highly in the past
  - only the items that have a high degree of similarity to whatever the user's preference are would be recommened

# Content-based Recommendation

- Approaches
  - Information Filtering
    - Routing
    - the queries remain relatively static while new documents come into the system
    - User profile is kept to filter information
    - Push, rather than pull
  - Classification
    - Two way classification problem
    - Learning the user's preference from training samples (users preference in the past)

# Information Filtering

- User profiles
  - contain information about users' taste, preferences , and needs
  - can be elicited from users
    - explicitly, e.g. questionnaries
    - implicitly, learned from transactional behavior over time
- utility function $u(c, s)$

  $u(c,s)=score(ContentBasedProfile(c), Content(s))$

  $ContentBasedProfile(c)$: the profile of user

  $Content(s)$: the profile of item

# Information Filtering (cont.)

- Profile: vector space model (a vector of weights)

| | data | structure | video | multimedia | audio | MPEG |
|---|---|---|---|---|---|---|
| item = | 6 | 4 | 10 | 0 | 0 | 2 |
| user = | 10 | 8 | 0 | 8 | 0 | 0 |

- User profile
  - computed from individually rated content vectors
  - averaging approach: average vector from individual content vectors

# Limitations of Content-based Recommendations

- Limited Content Analysis
  - feature extraction
  - two difference items are represented by the same set of features, they are indistinguishable
- Overspecialization
  - limited to being recommended item that are similar to those already rated
  - too similar items (diversity of recommendations)
- New User Problem
  - new user having very few ratings cold start

# Collaborative Recommendation

# Collaborative Recommendation

- Collaborative recommendation
  - try to predict the utility of items for a particular user based on the items previously rated by other users
  - the utility $u(c, s)$ of item $s$ for user $c$ is estimated based on the utilities $u(c_j, s)$ assigned to item $s$ by those users $c_j \in C$ who are similar to user $c$.
- Approaches
  - memory-based
  - model-based

# Approaches of Collaborative Recommendation

- Memory-based
  - prediction online based on user preference information
  - model learning is not required
  - user-based CF collaborative filtering
- Model-based
  - based on offline preprocessing or model-learning
  - at run-time, only the learned model is used to make predictions
  - models are updated (retrained) periodically
  - model-building and updating can be computationally expensive
  - item-based CF

# Memory-based

- memory-based (heuristic-based)
  - heuristics that make rating predictions based on the entire collection of previously rated items by the users
  - unknown rating $r_{c,s}$ for a user $c$ & item $s$

$$r_{c,s} = \underset{c' \in C'}{aggr}\, r_{c',s}$$

aggregation fnc, often weighted sum based on similarity

- Problem: different users may use the rating scale differently
  - adjusted weighted sum
  - preference-based filtering: relative preference

# Memory-based (cont.)

- User-centric approach
  - Nearest neighbor users
    - based on ratings of items that both users have rated
    - approaches
      - correlation-based
      - cosine-based
  - Graph-theoretic approach: IMB IRA
- Item-centric approach
  - Compute similarity between items
  - Computes the prediction on an item $i$ for a user $u$ by computing the sum of the ratings given by the user on the items similar to $i$

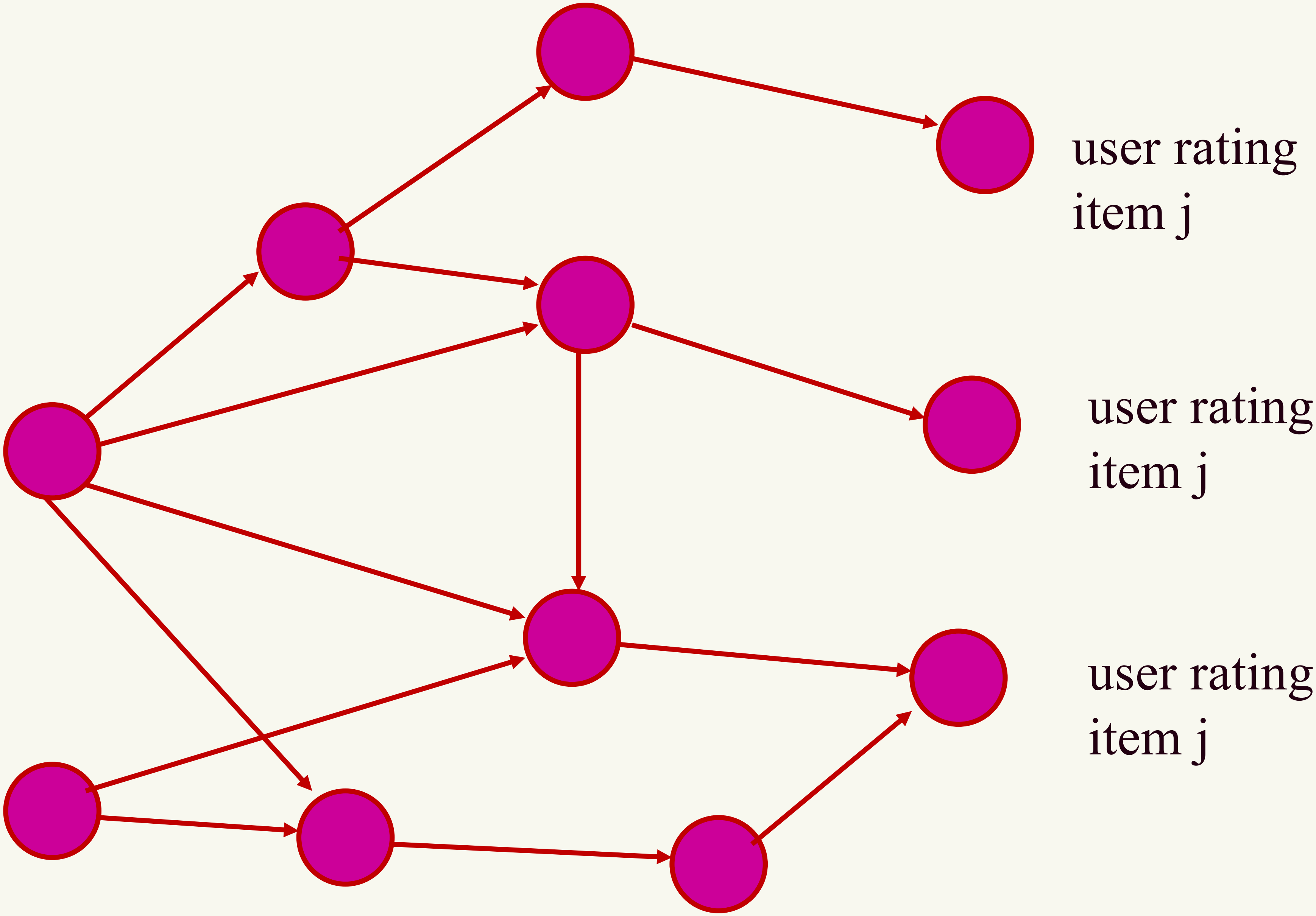
# Memory-based: IBM IRA

- IRA: rating-based recommendation engine
- Concept
  - horting: customer 1 horts customer 2 provided number of items both customers have rated in common normalized by number of items rated by customer 1 exceeds some threshold

    * if customer1 horts customer 2, it doesn't follow that customer 2 horts customer 1 (this why not cohort)
  - predictability: customer 2 predict customer 1 if ① customer 1 hort customer 2 & ② there exists a *linear transformation* for which the *transformed rating* of customer2 and rating of customer 1 are close
  - shortest path in a directed graph

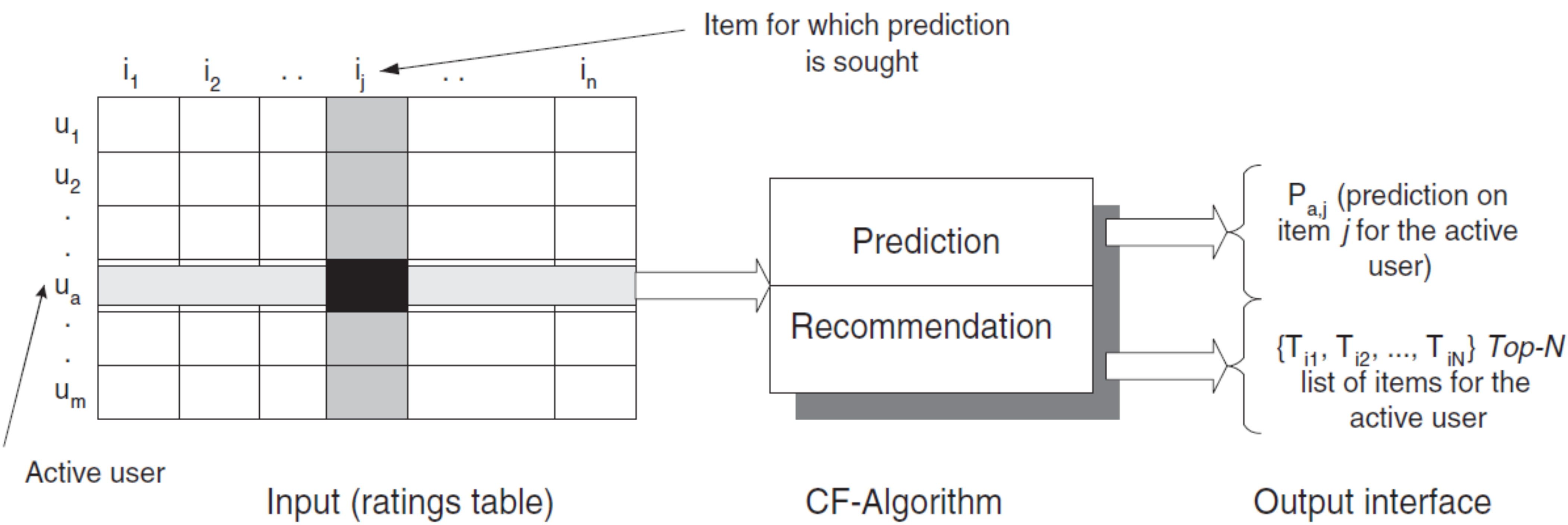
# Memory-based: IBM IRA (cont.)

- IRA maintains 2 data structure
  - For each item , maintains an inverted index
  - Maintains a directed graph
    - Vertex: customer
    - Directed edge <i, j>: customer j predicts customer i
- If a customer updates rating of items
  - Find the set of customer horted by this customer using inverted index
  - Find the subset of customer who predict the target customer
  - Shortest path from any customer in the predictive subset to this customer is computed


user rating
item j
user rating
item j
user rating
item j

# Collaborative Recommendation (cont.)

- Process of collaborative recommendation



↗ Collaborative Filtering

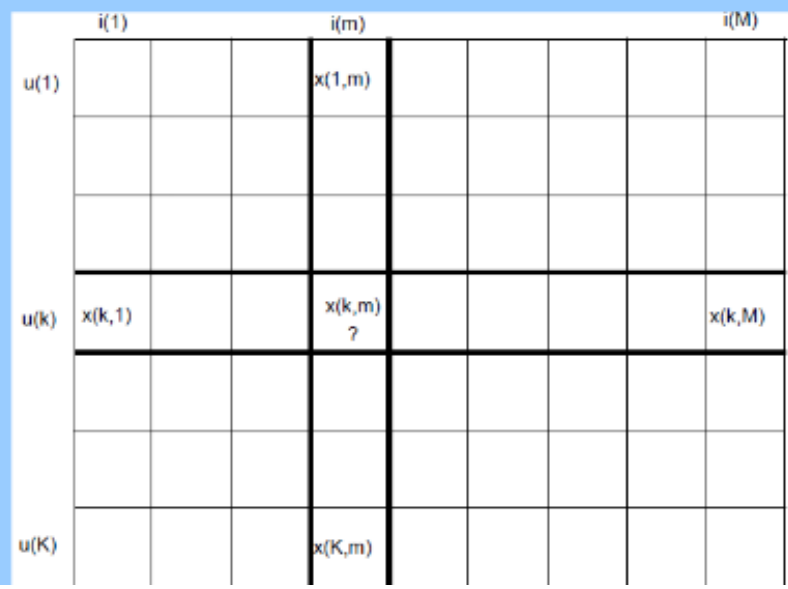
# User based CF algorithm

- The User x Item Matrix:

| | Shrek | Snow-white | Superman |
|---|---|---|---|
| Alice | Like | Like | Dislike |
| Bob | ? | Dislike | Like |
| Chris | Like | Like | Dislike |
| Jon | Like | Like | ? |

Shall we recommend Superman for John?

Jon's taste is similar to both Chris and Alice tastes → Do not recommend Superman to Jon




(ref: WWW 2011 Tutorial)

# User based CF algorithm (cont)

- Voting: $v_{ij}$ corresponds to the vote of user *i* for item j

•The mean vote for user i:
$$\overline{v_i} = \frac{1}{|Ii|} \sum_{i_j \in I_i} v_{ij}$$

- $I_i$ is the set of items on which user *i* voted

•Predictive vote:
$$p_{i,j} = \overline{v_i} + \kappa \sum_{k=1}^{n} w(i,k)(v_{kj} - \overline{v_k})$$

- *w(i,k) -the similarity between $u_i$ and $u_k$*
- k – a normalization factor

$$\kappa = \left(\sum w(i,k)\right)^{-1}$$



(ref: WWW 2011 Tutorial)

**Predictive Vote Formula**

$$p_{i,j} = \overline{v_i} + k \sum_{k=1}^{n} w(i,k)(v_{k,j} - \overline{v_k})$$

**Components:**

1. $\overline{v_i}$**:**
   - The mean vote (average rating) of user $i$.
   - Formula:

$$\overline{v_i} = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j}$$

Where $I_i$ is the set of items rated by user $i$.

2. $w(i,k)$**:**
   - The similarity between user $i$ and another user $k$.
   - Measures how similar the preferences of the two users are (e.g., using Pearson correlation or cosine similarity).
3. $(v_{k,j} - \overline{v_k})$**:**
   - The deviation of user $k$'s rating for item $j$ ($v_{k,j}$) from their average rating ($\overline{v_k}$).
   - Captures how much user $k$ rates item $j$ relative to their typical ratings.
4. $k$**:**
   - A normalization factor to ensure the predictive vote is properly scaled.
   - Formula:

$$k = \left( \sum_{k=1}^{n} w(i,k) \right)^{-1}$$

# Cosine based similarity between users

$$w(u,v)=\frac{\sum_{i\in I} r_{u,i} r_{v,i}}{\sqrt{\sum_{i\in I} {r_{u,i}}^2}\sqrt{\sum_{i\in I} {r_{v,i}}^2}}$$

# Pearson based similarity between users

$$w(u,v)=\frac{\sum_{i\in I}(r_{u,i}-\overline{r_u})(r_{v,i}-\overline{r_v})}{\sqrt{\sum_{i\in I}(r_{u,i}-\overline{r_u})^2}\sqrt{\sum_{i\in I}(r_{v,i}-\overline{r_v})^2}}$$



(ref: WWW 2011 Tutorial)

# Item-Based Nearest Neighbor Algorithms

- The transpose of the user-based algorithms
  - generate predictions based on similarities between items
  - The prediction for an item is based on the user's ratings for similar items

| | Shrek | Snow-white | Superman |
|---|---|---|---|
| Alice | Like | Like | Dislike |
| Bob | ? | Dislike | Like |
| Chris | Like | Like | Dislike |
| Jon | Like | Like | ? |

Bob dislikes Snow-white (which is similar to Shrek) → do not recommend Shrek to Bob

$$p_{i,j} = \kappa \sum_{k=1}^{m} w(k, j) \cdot v_{i,k}$$

- *traverse over all m items rated by user i and measure their rate , averaged by their similarity to the predicted item*
- *w(k,j)* is a measure of item similarity - usually the cosine measure
- Average correcting is not needed because the component ratings are all from the same target user



(ref: WWW 2011 Tutorial)

# Dimensionality Reduction Algorithms

- Reduce domain complexity by mapping the item space to a smaller number of underlying "dimensions"
  - represent the latent topics or tastes present in those items
  - improve accuracy in predicting ratings for the most part
  - reduce run-time performance needs and lead to larger numbers of co-rated dimensions
- Popular techniques: Singular value decomposition and Principal Component Analysis
  - Require an extremely expensive offline computation step to generate the latent dimensional space

(ref: WWW 2011 Tutorial)

# Recommendation Based on Association Rule

| | Item1 | Item2 | Item3 | Item4 | Item5 |
|---|---|---|---|---|---|
| Alice | 1 | 0 | 0 | 0 | ? |
| User1 | 1 | 0 | 1 | 0 | 1 |
| User2 | 1 | 0 | 1 | 0 | 1 |
| User3 | 0 | 0 | 0 | 1 | 1 |
| User4 | 0 | 1 | 1 | 0 | 0 |

- **Simplest approach**
  - transform 5-point ratings into binary ratings (1 = above user average)
- **Mine rules such as**
  - Item1 → Item5
    - support (2/4), confidence (2/2) (without Alice)
- **Make recommendations for Alice (basic method)**
  - Determine "relevant" rules based on Alice's transactions (the above rule will be relevant as Alice bought Item1)
  - Determine items not already bought by Alice
  - Sort the items based on the rules' confidence values

| | Item1 | Item2 | Item3 | Item4 | Item5 |
|---|---|---|---|---|---|
| Alice | 5 | 3 | 4 | 4 | ? |
| User1 | 3 | 1 | 2 | 3 | 3 |
| User2 | 4 | 3 | 4 | 3 | 5 |
| User3 | 3 | 3 | 1 | 5 | 4 |
| User4 | 1 | 5 | 5 | 2 | 1 |

$$M = U \times \Sigma \times V^T$$

- **SVD:** $M_k = U_k \times \Sigma_k \times V_k^T$

| $U_k$ | Dim1 | Dim2 |
|---|---|---|
| **Alice** | 0.47 | -0.30 |
| **Bob** | -0.44 | 0.23 |
| **Mary** | 0.70 | -0.06 |
| **Sue** | 0.31 | 0.93 |

| $V_k^T$ | Terminator | Die Hard | Twins | Eat Pray Love | Pretty Woman |
|---|---|---|---|---|---|
| **Dim1** | -0.44 | -0.57 | 0.06 | 0.38 | 0.57 |
| **Dim2** | 0.58 | -0.66 | 0.26 | 0.18 | -0.36 |

| $\Sigma_k$ | Dim1 | Dim2 |
|---|---|---|
| **Dim1** | 5.63 | 0 |
| **Dim2** | 0 | 3.23 |

- **Prediction:** $\hat{r}_{ui} = \bar{r}_u + U_k(Alice) \times \Sigma_k \times V_k^T(EPL)$

**= 3 + 0.84 = 3.84**

# Limitations of Collaborative Recommendations

- New Item Problem
  - until the new item is rated by a substantial number of users, the recommender system would not be able to recommend it
- Sparsity
  - the number of ratings already obtained is usually very small compared to the number of ratings that need to be predicted.