# CSCI567 Machine Learning (Fall 2017)

Parisa Mansourifard

U of Southern California

Lecture on Nov. 21, 2017

# Outline

1. Recommender Systems: Introduction
2. Content-based recommendations
3. Collaborative Filtering
4. Hybrids

# Outline

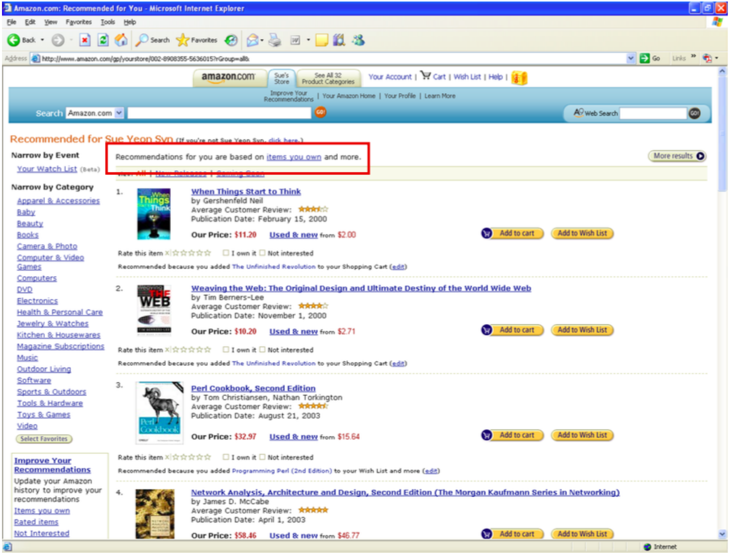
1 Recommender Systems: Introduction

2 Content-based recommendations

3 Collaborative Filtering

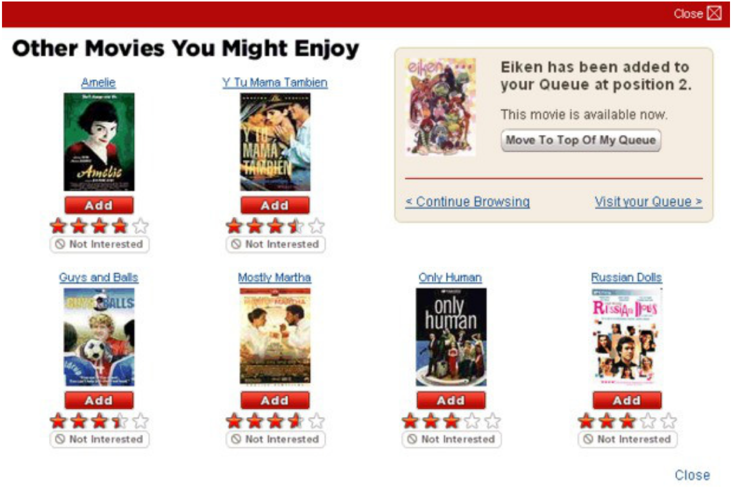
4 Hybrids

# Example: Amazon.com



# Example: Twitter

# Example: Netflix



# Example: Movie recommendation

Training data

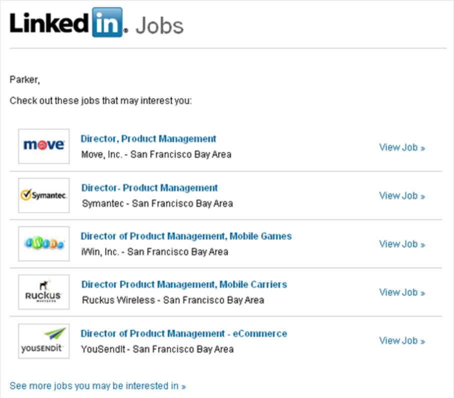
| user | movie | score |
|---|---|---|
| 1 | 21 | 1 |
| 1 | 213 | 5 |
| 2 | 345 | 4 |
| 2 | 123 | 4 |
| 2 | 768 | 3 |
| 3 | 76 | 5 |
| 4 | 45 | 4 |
| 5 | 568 | 1 |
| 5 | 342 | 2 |
| 5 | 234 | 2 |
| 6 | 76 | 5 |
| 6 | 56 | 4 |

Test data

| user | movie | score |
|---|---|---|
| 1 | 62 | ? |
| 1 | 96 | ? |
| 2 | 7 | ? |
| 2 | 3 | ? |
| 3 | 47 | ? |
| 3 | 15 | ? |
| 4 | 41 | ? |
| 4 | 28 | ? |
| 5 | 93 | ? |
| 5 | 74 | ? |
| 6 | 69 | ? |
| 6 | 83 | ? |

# Recommender Systems: Other Examples

- Recommendation of groups, jobs or people on LinkedIn
- Friend recommendation and ad personalization on Facebook
- News recommendation at Forbes.com
- etc.



# Why using Recommender Systems?

- Value for the customer
    - Find things that are interesting
    - Narrow down the set of choices
    - Help me explore the space of options
    - Discover new things
    - Entertainment, ...
- Value for the provider
    - Additional and probably unique personalized service for the customer
    - Increase sales, click trough rates, conversion etc.
    - Opportunities for promotion, persuasion
    - Obtain more knowledge about customers

# Value of Recommender Systems

- Netflix: 2/3 of the movies watched are recommended
- Google News: recommendations generate $38\%$ more clickthrough
- Amazon: $35\%$ sales from recommendations
- Choicestream: $28\%$ of the people would buy more music if they found what they liked.

# Recommender Systems: Problem

Estimate a utility function that automatically predicts how a user will like an item.

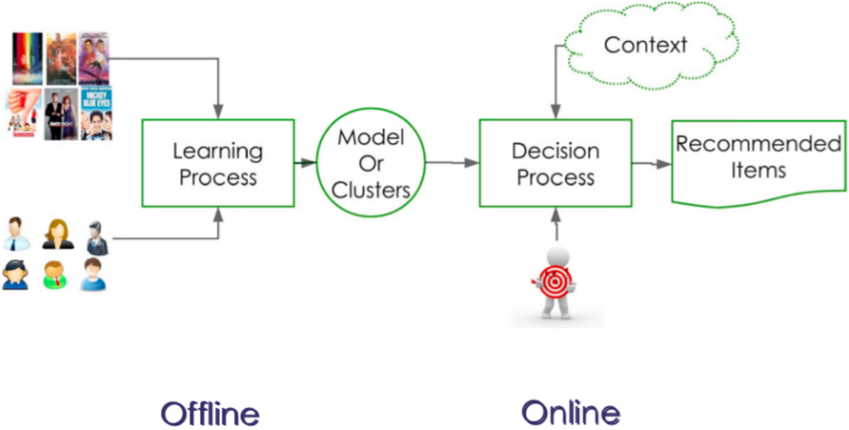
Based on:

- Past behavior
- Relations to other users
- Item similarity
- Context
- ...

# Recommender Systems: Problem

- Let $U$ be set of all users and let $I$ be set of all possible recommendable items
- Let $F$ be a utility function measuring the usefulness of item $i$ to user $u$, i.e., $F : I \times U \rightarrow R$, where $R$ is a totally ordered set.
- For each user $u \in U$, we want to choose items $i \in I$ that maximize $F$.

# Recommender Systems: Two-step Process



# Recommender Systems: Approaches

Popular approaches:

- Content-based recommendations:
  - The user will be recommended items based on profile information or similar to the ones the user preferred in the past
- Collaborative filtering (or collaborative recommendations):
  - The user will be recommended items that people with similar tastes and preferences liked in the past
- Hybrids:
  - Combine collaborative and content-based methods

# Outline


1. Recommender Systems: Introduction
2. Content-based recommendations
3. Collaborative Filtering
4. Hybrids

# Content-based recommendations

- Recommend items that matches the User Profile.
- The Profile is based on items user has liked in the past or explicit interests that he defines.
- A content-based recommender system matches the profile of the item to the user profile to decide on its relevancy to the user.

# Outline

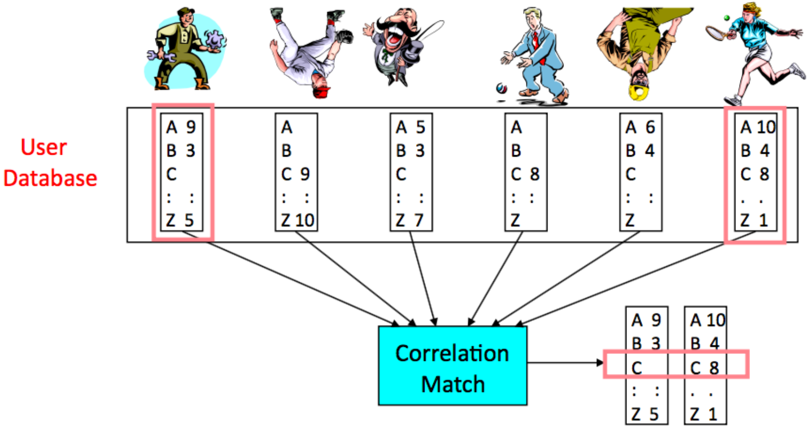
1 Recommender Systems: Introduction

2 Content-based recommendations

3 Collaborative Filtering

4 Hybrids

# Collaborative Filtering



# Collaborative Filtering

- Collaborative filtering (CF):
  - k-nearest neighbor (kNN)
  - matrix factorization
- The most prominent approach to generate recommendations: Used by large, commercial e-commerce sites
- Well-understood, various algorithms and variations exist
- Applicable in many domains (book, movies, DVDs, ..)
- Approach: use the wisdom of the crowd to recommend items
- Basic assumption and idea
  - Users give ratings to catalog items (implicitly or explicitly)
  - Customers who had similar tastes in the past, will have similar tastes in the future

# Collaborative Filtering: User-based Model

GroupLens: an open architecture for collaborative filtering of netnews, P. Resnick et al., ACM CSCW, 1994.

- Goal: given an active user (Alice) and an item $i$ not yet seen by Alice, the goal is to estimate Alices rating for this item.
- Basic idea: People who agreed in their subjective evaluations in the past are likely to agree again in the future
- Algorithm: find a set of users (peers) who liked the same items as Alice in the past and who have rated item $p$, e.g. the average of their ratings to predict, if Alice will like item $p$; do this for all items (Alice has not seen and recommend the best-rated)

| | 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 |

# Collaborative Filtering: Important Questions

- How do we measure similarity?
- How many neighbors should we consider?
- How do we generate a prediction from the neighbors ratings?

# Collaborative Filtering: Important Questions

- How do we measure similarity?
  Answer: Pearson correlation

$$sim(a,b) = \frac{\sum_p (r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_p (r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_p (r_{b,p} - \bar{r}_b)^2}}$$

- How many neighbors should we consider?
  Answer: Use similarity threshold or fixed number of neighbors.

# Collaborative Filtering: Important Questions

- How do we generate a prediction from the neighbors ratings?
  Answer:

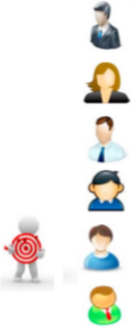
$$pred(a,p) = \bar{r}_a + \frac{\sum_{b \in N(a)} sim(a,b) * (r_{b,p} - \bar{r}_b)}{\sum_{b \in N(a)} sim(a,b)}$$

a. Calculate whether the neighbors ratings for the unseen item $i$ are higher or lower than their average

b. Combine the rating differences using the similarity as a weight

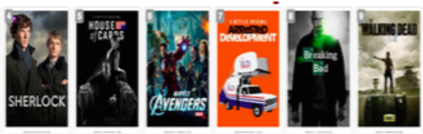
c. Add/subtract the neighbors bias from the active users average and use this as a prediction

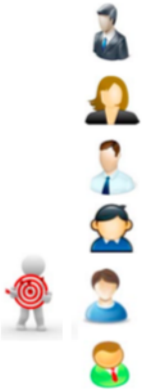
# User-based CF: Example

| | | | | | | sim(u,v) |
|---|---|---|---|---|---|---|
| 2 | | | 4 | 5 | | NA |
| 5 | | 4 | | | 1 | |
| | | 5 | | 2 | | |
| | 1 | | 5 | | 4 | |
| | | 4 | | | 2 | |
| 4 | 5 | | 1 | | | NA |

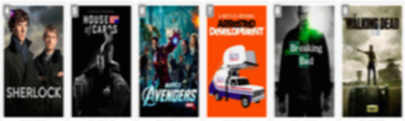
# User-based CF: Example



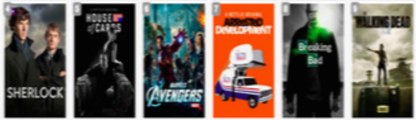
| | | | | | | sim(u,v) |
|---|---|---|---|---|---|---|
| 2 | | | 4 | 5 | | NA |
| 5 | | 4 | | | 1 | 0.87 |
| | | 5 | | 2 | | |
| | 1 | | 5 | | 4 | |
| | | 4 | | | 2 | |
| 4 | 5 | | 1 | | | NA |

# User-based CF: Example



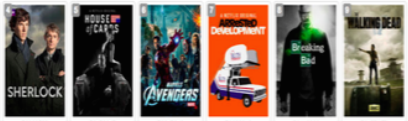

| | | | | | | sim(u,v) |
|---|---|---|---|---|---|---|
| 2 | | | 4 | 5 | | NA |
| 5 | | 4 | | | 1 | 0.87 |
| | | 5 | | 2 | | 1 |
| | 1 | | 5 | | 4 | |
| | | 4 | | | 2 | |
| 4 | 5 | | 1 | | | NA |

# User-based CF: Example




| | | | | | | sim(u,v) |
|---|---|---|---|---|---|---|
| 2 | | | 4 | 5 | | NA |
| 5 | | 4 | | | 1 | 0.87 |
| | | 5 | | 2 | | 1 |
| | 1 | | 5 | | 4 | -1 |
| | | 4 | | | 2 | |
| 4 | 5 | | 1 | | | NA |

# User-based CF: Example




| | | | | | | sim(u,v) |
|---|---|---|---|---|---|---|
| 2 | | | 4 | 5 | | NA |
| 5 | | 4 | | | 1 | 0.87 |
| | | 5 | | 2 | | 1 |
| | 1 | | 5 | | 4 | -1 |
| 3.51* | 3.81* | 4 | 2.42* | 2.48* | 2 | |
| 4 | 5 | | 1 | | | NA |

# User-based CF: Challenges

- Sparsity evaluation of large item sets, users purchases are under $1\%$.
- Difficult to make predictions based on nearest neighbor algorithms $\rightarrow$ Accuracy of recommendation may be poor.
- Scalability - Nearest neighbor require computation that grows with both the number of users and the number of items.
- Poor relationship among like minded but sparse-rating users.
- Solution : usage of latent models to capture similarity between users and items in a reduced dimensional space

# Collaborative Filtering: Item-based Model

Item-based collaborative filtering recommendation algorithms, B. Sarwar et al., WWW 2001.

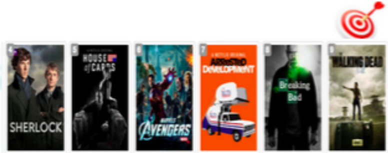
- Scalability issues arise with user-based model if there are many more users than items ($m >> n, m = \#$ of users, $n = \#$ of items). e.g. amazon.com.
  - Space complexity is $O(m^2)$ when pre-computed;
  - Time complexity for computing Pearson is $O(m^2 n)$.
- High sparsity leads to few common ratings between two users
- Basic idea: Item-based CF exploits relationships between items first, instead of relationships between users.

| | 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 |

# Collaborative Filtering: Item-based Model

- Item-based filtering does not solve the scalability problem itself
- Pre-processing approach by Amazon.com (in 2003)
  (1) Calculate all pair-wise item similarities in advance
  (2) The neighborhood to be used at run-time is typically rather small, because only items are taken into account which the user has rated
  (3) Item similarities are supposed to be more stable than user similarities
- Memory requirements: Up to $N^2$ pair-wise similarities to be memorized (N $=$ number of items) in theory; In practice, this is significantly lower (items with no co-ratings)
- Minimum threshold for co-ratings (items, which are rated at least by n users)
- Limit the size of the neighborhood (might affect recommendation accuracy)

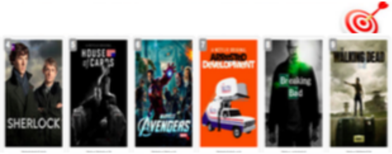
# Item-based CF: Example




| | | | | | |
|---|---|---|---|---|---|
| 2 | | | 4 | 5 | |
| 5 | | 4 | | | 1 |
| | | 5 | | 2 | |
| | 1 | | 5 | | 4 |
| | | 4 | | | 2 |
| 4 | 5 | | 1 | | |

sim(i,j) -1

# Item-based CF: Example

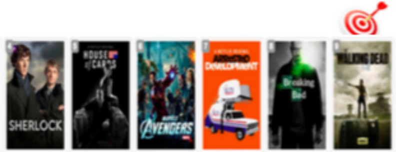



| | | | | | | |
|---|---|---|---|---|---|---|
| | 2 | | | 4 | 5 | |
| | 5 | | 4 | | | 1 |
| | | | 5 | | 2 | |
| | | 1 | | 5 | | 4 |
| | | | 4 | | | 2 |
| | 4 | 5 | | 1 | | |
| sim(i,j) | -1 | -1 | | | | |

# Item-based CF: Example


| | | | | | | |
|---|---|---|---|---|---|---|
| | 2 | | | 4 | 5 | |
| | 5 | | 4 | | | 1 |
| | | | 5 | | 2 | |
| | | 1 | | 5 | | 4 |
| | | | 4 | | | 2 |
| | 4 | 5 | | 1 | | |
| sim(i,j) | -1 | -1 | 0.86 | | | |

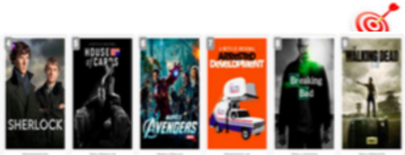
# Item-based CF: Example




| | | | | | |
|---|---|---|---|---|---|
| 2 | | | 4 | 5 | |
| 5 | | 4 | | | 1 |
| | | 5 | | 2 | |
| | 1 | | 5 | | 4 |
| | | 4 | | | 2 |
| 4 | 5 | | 1 | | |

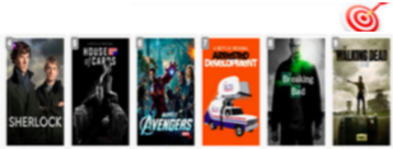
| sim(i,j) | -1 | -1 | 0.86 | 1 | | |
|---|---|---|---|---|---|---|

# Item-based CF: Example




| | | | | | | |
|---|---|---|---|---|---|---|
| | 2 | | | 4 | 5 | |
| | 5 | | 4 | | | 1 |
| | | | 5 | | 2 | |
| | | 1 | | 5 | | 4 |
| | | | 4 | | | 2 |
| | 4 | 5 | | 1 | | |
| sim(i,j) | -1 | -1 | 0.86 | 1 | NA | |

# Item-based CF: Example




| | | | | | | |
|---|---|---|---|---|---|---|
| | 2 | | | 4 | 5 | 2.94* |
| | 5 | | 4 | | | 1 |
| | | | 5 | | 2 | 2.48* |
| | | 1 | | 5 | | 4 |
| | | | 4 | | | 2 |
| | 4 | 5 | | 1 | | 1.12* |
| sim(i,j) | -1 | -1 | 0.86 | 1 | NA | |

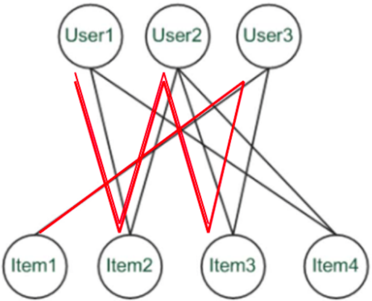
# Performance Implications

- Bottleneck - Similarity computation.
- Time complexity, highly time consuming with millions of users and items in the database.
  - Isolate the neighborhood generation and predication steps.
  - off-line component / model similarity computation, done earlier and stored in memory.
  - on-line component prediction generation process.

# Collaborative Filtering: Graph-based Model

Spreading activation (sketch)

- Use paths of lengths $>= 3$ to recommend items
- Length 3: Recommend Item 3 to User 1
- Length 5: Item 1 also recommendable



# Collaborative Filtering: Dimension Reduction

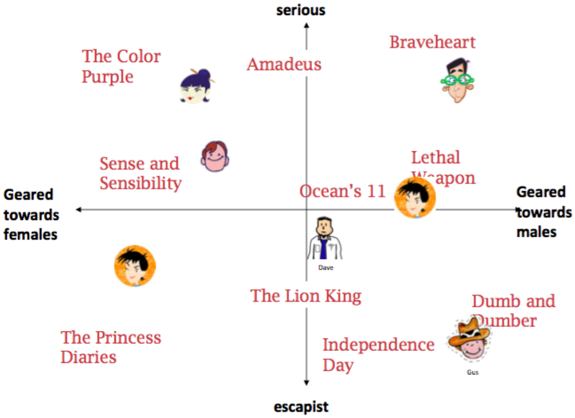
Application of Dimensionality Reduction in Recommender System, B. Sarwar et al., WebKDD Workshop, 2000.

Basic idea: Trade more complex offline model building for faster online prediction generation

- Singular Value Decomposition for dimensionality reduction of rating matrices
- Captures important factors/aspects and their weights in the data factors can be genre, actors, or some semantics
- Assumption that k dimensions capture the signals and filter out noise

# Collaborative Filtering: Dimension Reduction

Application of Dimensionality Reduction in Recommender System, B. Sarwar et al., WebKDD Workshop, 2000.



# Collaborative Filtering: Dimension Reduction

Application of Dimensionality Reduction in Recommender System, B. Sarwar et al., WebKDD Workshop, 2000.

- **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 |

| $V_k^T$ |  |  |  |  |  |
|---|---|---|---|---|---|
| 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**

# Collaborative Filtering: Latent Class Models

Latent Class Models for Collaborative Filtering, Hofmann and Puzieha, IJCAI 1999.

- Input: user $u$ and item $i$ with rating $v$
- One example of the latent class model: the vote on an item is independent of the users identity given the true value of the latent variable $z$, i.e.,

$$P(v|u,i) = \sum_z P(v|u,i,z)P(z|u) = \sum_z P(v|i,z)P(z|u).$$

# Collaborative Filtering: Hybrid Approach

Factorization meets the neighborhood: a multifaceted collaborative filtering model, Y. Koren, ACM SIGKDD, 2008.

Basic idea: Combination of different approaches in one prediction single function

- Baseline estimates: $v_{u,i} = \mu + b_u + b_i$. The resulting optimization problem is:

$$\min_{b^*} \sum_{(u,i)\in\mathcal{K}} (v_{u,i} - \mu - b_u - b_i)^2 + \lambda_1 (\sum_{u'} b_{u'}^2 + \sum_{v'} b_{v'}^2)$$

- Baseline + Latent model: $v_{u,i} = \mu + b_u + b_i + p_u^T q_i$. The resulting optimization problem is:

$$\min_{b^*,p^*,q^*} \sum_{(u,i)\in\mathcal{K}} (v_{u,i} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda_1 (\sum_{u'} b_{u'}^2 + \sum_{v'} b_{v'}^2)$$
$$+ \lambda_2 (||p||_2 + ||q||_2)$$

# Evaluation Metrics

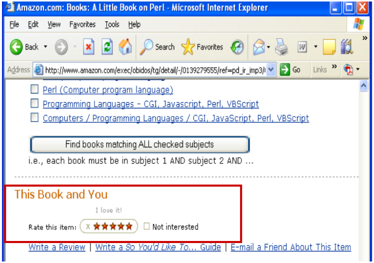
- Novelty: The ability of a recommender system to recommend items that the user was not already aware of.
- Serendipity: Users are given recommendations for items that they would not have seen given their existing channels of discovery. A measure of how surprising the recommendations are
- Coverage: The percentage of the items known to the recommender system for which it can generate predictions.
- Learning Rate: How quickly the CF system becomes an effective predictor of taste as data begins to arrive
- Confidence: Ability to evaluate the likely quality of its predictions
- User Satisfaction: By surveying the users or measuring retention and use statistics

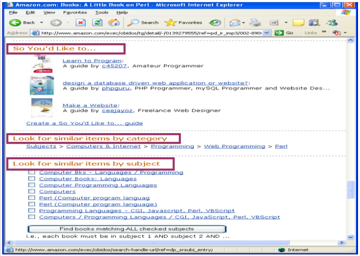
# Evaluation Metrics

- Accuracy
  - Predict accuracy: The ability of a recommender system to predict a users rating for an item
    - Mean absolute error (MAE)
  - Rank accuracy
    - Precision: percentage of items in a recommendation list that the user would rate as useful
    - Half-life utility: the utility of a recommendation list to a user
      Assumption: the likelihood that a user examines a recommended object decays exponentially with the objects ranking

# Practical Issues: Ratings

- Explicit ratings:
  - Users rate themselves for an item
  - Most accurate descriptions of a users preference
  - Challenging in collecting data
- Implicit ratings
  - Observations of user behavior
  - Can be collected with little or no cost to user
  - Ratings inference may be imprecise.



# Practical Issues: Cold start

Not enough data in order to make accurate recommendations

- New user
  - Rate some initial items
  - Non-personalized recommendations - Describe tastes
  - Demographic info.
- New Item
  - Non-CF : content analysis, metadata

# Practical Issues: Others

- Scalability:
  - Usually, there are millions of users and products
  - Large amount of computation power is necessary
- Sparsity:
  - The number of items is extremely large
  - The most active users will only have rated a small subset of the overall database
  - Even the most popular items have a very few ratings
- Privacy and Trust:
  - Users profiles, Personalized information
  - Recommender system may break trust when malicious users give ratings that are not representative of their true preferences.

# Collaborative Filtering: Summary

- Pros: well-understood, works well in some domains, no knowledge engineering required
- Cons: requires user community, sparsity problems, no integration of other knowledge sources, no explanation of results
- What is the best CF method? Differences between methods could be very small with limited data
- How to evaluate the prediction quality? MAE / RMSE
  - What does an MAE of 0.7 actually mean?

# Outline

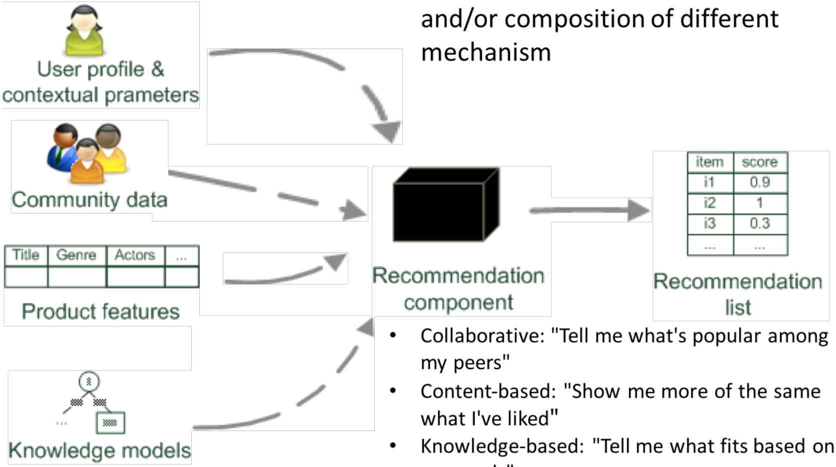
1 Recommender Systems: Introduction

2 Content-based recommendations

3 Collaborative Filtering

4 Hybrids

# Hybrid recommendation




Hybrid: combinations of various inputs and/or composition of different mechanism

- Collaborative: "Tell me what's popular among my peers"
- Content-based: "Show me more of the same what I've liked"
- Knowledge-based: "Tell me what fits based on my needs"

# Credits

Parts of this lecture note is based on the slides of:

- Prof. Yan Liu
- Narges Razavian@ csail.mit
- Alex Smola@CMU
- Yahuda Koren@Yahoo labs
- Bing Liu@UIC
- Xaviar Amatriain@ Netflix