IR: Information Retrieval

FIB, Master in Innovation and Research in Informatics

Slides by Marta Arias, José Balcázar, Ricard Gavaldá
Department of Computer Science, UPC

Fall 2016

http://www.cs.upc.edu/~ir

9. Recommender Systems

Outline

1. Recommending: What and why?
2. Collaborative filtering approaches
3. Content-based approaches
4. Recommending in social networks

(Slides based on a presentation by Irena Koprinska (2012), with thanks)

Recommender Systems

Recommend items to users

- Which digital camera should I buy?
- What is the best holiday for me?
- Which movie should I rent?
- Which websites should I follow?
- Which book should I buy for my next holiday?
- Which degree and university are the best for my future?

Sometimes, items are people too:

- Which Twitter users should I follow?
- Which writers/bloggers should I read?

Why?

How do we find good items?

- Friends
- Experts
- Searchers: Content-based and link based
- ...

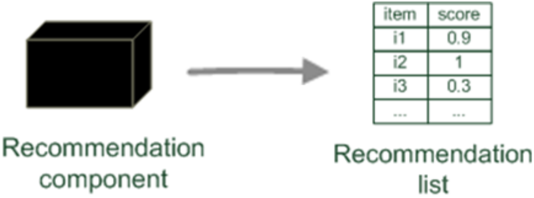
Why?

The paradox of choice:

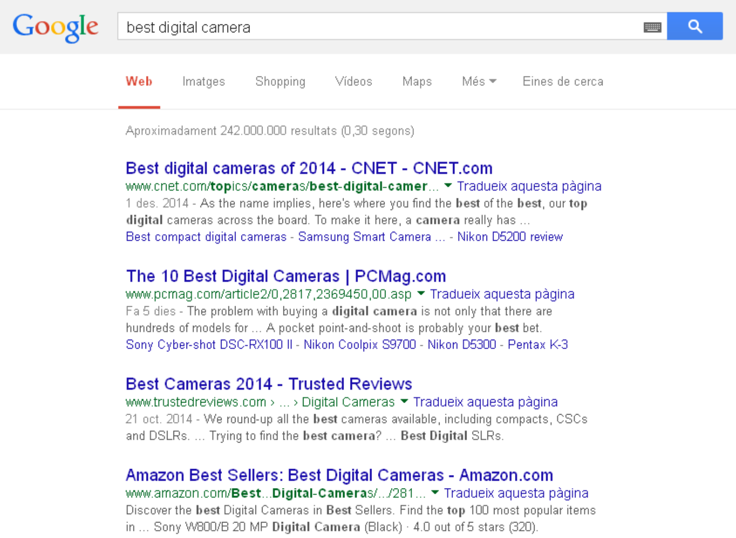
- 4 types of jam or 24 types of jam?

Why?

- The web has become the main source of information
- Huge: Difficult to find “best” items - can’t see all
- Recommender systems help users to find products, services, and information, by predicting their relevance



Recommender Systems vs. Search Engines



How to recommend

The recommendation problem:

Try to predict items that will interest this user

- Top-N items (ranked)
- All interesting items (few false positives)
- A sequence of items (music playlist)

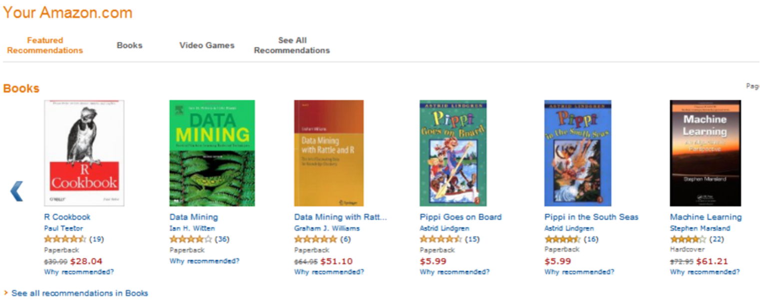
Based on what information?

User profiles

Ask the user to provide information about him/herself and interests

But:

- People won't bother
- People may have multiple profiles



Ratings

- Explicit (1..5, “like”)
  - hard to obtain many
- Implicit (clicks, page views, downloads)
  - unreliable
  - e.g. did the user like the book he bought?
  - did s/he buy it for someone else?

Methods

- Baseline: Recommend most popular items
- Collaborative filtering
- Content-based
- Hybrid

Collaborative Filtering

- Trusts wisdom of the crowd
- Input: a matrix of user-to-item ratings, an active user
- Output: top-N recommendations for active user

Main CF methods

- Nearest neighbors:
  - user-to-user: uses the similarity between users
  - item-to-item: uses the similarity between items

- Others:
  - Matrix factorization: maps users and items to a joint factor space
  - Clustering
  - Probabilistic (not explained)
  - Association rules (not explained)
  - …

User-to-user CF: Basic idea

Recommend to you what is rated high by people with ratings similar to yours

- If you and Joe and Jane like band X,
- and if you and Joe and Jane like band Y,
- and if Joe and Jane like band Z, which you never heard about,
- then band Z is a good recommendation for you

Nearest neighbors

User-to-user:

1. Find k nearest neighbors of active user
2. Find set C of items bought by these k users, and their frequencies
3. Recommend top-N items in C that active user has not purchased

Step 1 needs “distance” or “similarity” among users

User-to-user similarity

	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

Correlation as similarity:

- Users are more similar if their common ratings are similar
- E.g. User 2 most similar to Alice

User-to-user similarity

$r_{i,s}$: rating of item s by user i
a, b: users
S: set of items rated both by a and b
$\bar{r}_a$, $\bar{r}_b$: average of the ratings by a and b

$$sim(a,b) = \frac{\sum_{s \in S}(r_{a,s} - \bar{r}_a) \cdot (r_{b,s} - \bar{r}_b)}{\sqrt{\sum_{s \in S}(r_{a,s} - \bar{r}_a)^2} \cdot \sqrt{\sum_{s \in S}(r_{b,s} - \bar{r}_b)^2}}$$

Cosine similarity or Pearson correlation

Combining the ratings

How will a like item s?

- Simple average among similar users b
- Average weighted by similarity of a to b
- Adjusted by considering differences among users

$$pred(a, s) = \bar{r}_a + \frac{\sum_b sim(a, b) \cdot (r_{b,s} - \bar{r}_b)}{\sum_b sim(a, b)}$$

Variations

- Number of co-rated items: Reduce the weight when the number of co-rated items is low
- Case amplification: Higher weight to very similar neighbors
- Not all neighbor ratings are equally valuable
  - E.g. agreement on commonly liked items is not so informative as agreement on controversial items
  - Solution: Give more weight to items that have a higher variance

Evaluation

Main metrics: Mean Average Error, average value of

$$|\, pred(a, s) - r_{a,s}|$$

Others:

- Diversity: Don't recommend Star Wars 3 after 1 and 2
- Surprise: Don't recommend "milk" in a supermarket
- Trust: For example, give explanations

Item-to-item CF

- Look at columns of the matrix
- Find set of items similar to the target one
- e.g., Items 1 and 4 seem most similar to Item 5

	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

- Use Alice's users' rating on Items 1 and 4 to rate Item 5
- Formulas can be as for user-to-user case

Can we precompute the similarities?

Rating matrix: a large number of items and a small number of ratings per user User-to-user collaborative filtering:

- Similarity between users is unstable (computed on few commonly rated items)
- $\rightarrow$ pre-computing the similarities leads to poor performance

Item-to-item collaborative filtering

- Similarity between items is more stable
- We can pre-compute the item-to-item similarity and the nearest neighbours
- Prediction involves lookup for these values and computing the weighed sum (Amazon does this)

Matrix Factorization Approaches

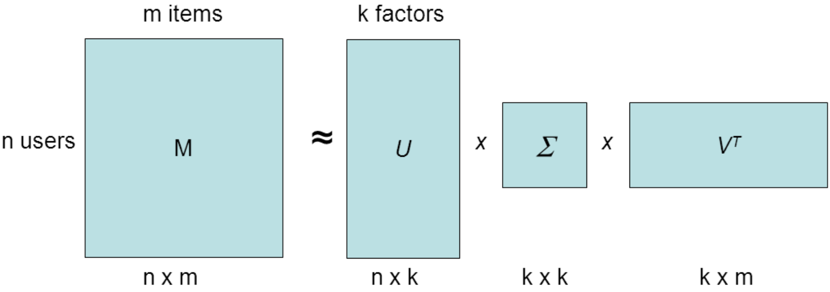
Singular Value Decomposition Theorem (SVD):

Theorem: Every $n \times m$ matrix M of rank K can be decomposed as $M = U\Sigma V^T$ where

- U is $n \times K$ and orthonormal
- V is $m \times K$ and normal
- Σ is $K \times K$ and diagonal

Furthermore, if we keep the $k < K$ highest values of Σ and zero the rest, we obtain the best approximation of M with a matrix of rank k

Matrix Factorization: Intepretation



- There are k latent factors - topics or explanations for ratings
- U tells how much each user is affected by a factor
- V tells how much each item is related to a factor
- Σ tells the weight of each different factor

Matrix Factorization: Method

Offline: Factor the rating matrix M as $U\Sigma V^T$

- This is costly computationally, and has a problem

Online: Given user a and item s, interpolate $M[a,s]$ from U, Σ, V

$$\begin{aligned} pred(a,s) &= U[a] \cdot \Sigma \cdot V^T[s] \\ &= \sum_k \Sigma_k \cdot U[a,k] \cdot V[k,s] \end{aligned}$$

= How much a is about each factor, times how much s is, summed over all latent factors

Matrix Factorization: Problem

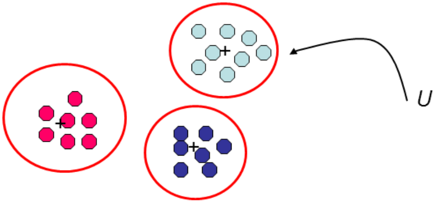
Matrix M has (many!) unknown, unfilled entries

Standard algorithms for finding SVD assume no missing values

$\rightarrow$ Formulate as a (costly) optimization problem: stochastic gradient descent, to minimize error on available ratings

State of the art method for CF, accuracywise

Clustering



- Cluster users according to their ratings (form homogeneous groups)
- For each cluster, form the vector of average item ratings
- For an active user U, assign to a cluster, return items with highest rates in cluster's vector

Simple and efficient, but not so accurate

CF - pros and cons

Pros:

- No domain knowledge: what “items” are, why users (dis)like them, not used

Cons:

- Requires user community
- Requires sufficient number of co-rated items
- The cold start problem:
  - *user:* what do we recommend to a new user (with no ratings yet)
  - *item:* a newly arrived item will not be recommended (until users begin rating it)
- Does not provide explanation for the recommendation

Content-based methods

Use information about the items and not about the user community

- e.g. recommend fantasy novels to people who liked fantasy novels in the past

What we need:

- Information about the content of the items (e.g. for movies: genre, leading actors, director, awards, etc.)
- Information about what the user likes (user preferences, also called user profile) - explicit (e.g. movie rankings by the user) or implicit
- Task: recommend items that match the user preferences

Content-based methods (2)

The rating prediction problem now:

Given an item described as a vector of (feature,value) pairs, predict its rating (by a fixed user)

Becomes a Classification / Regression problem, that can be addressed with Machine Learning methods (Naive Bayes, support vector machines, nearest neighbors, . . .)

Can be used to recommend documents (= tf-idf vectors) to users

Content-based: Pros and Cons

Pros:

- No user base required
- No item coldstart problem: we can predict ratings for new, unrated, items

  (the user coldstart problem still exists)

Cons:

- Domain knowledge required
- Hard work of feature engineering
- Hard to transfer among domains

Hybrid methods

For example:

- Compute ratings by several methods, separately, then combine
- Add content-based knowledge to CF
- Build joint model

Shown to do better than one method alone

Recommendation in Social Networks

Two meanings:

- Recommend to you “interesting people you should befriend / follow”
- Use your social network to recommend items to you

Common principle:

- We tend to like what our friends like (more than random)

The filter bubble

Potential problem pointed out by Eli Pariser:

> *As algorithms select information for us based on what they expect us to like, we become more separated from information that disagrees with our viewpoints, becoming isolated in our own cultural and ideological bubbles.*

Some studies disagree: recommendation does not distort that much results on a user-per-user basis

`http://www.ted.com/talks/eli_pariser_beware_online_filter_bubbles.html`

Further topics in RS

- Scalability, real-time
- Explanation
- Mobile, context-aware recommendations
- Diversity. Serendipity
- Two-way recommendations (e.g. dating sites)
- Team formation
- Group recommendations
- Privacy, robustness