Master on Foundations of Data Science

Recommender Systems

Collaborative Recommender Systems

Santi Seguí | 2016-2017

Generalization of Supervised Classification

Features

	x1	x2	x3	x4	x5	x6	x7
U1	13,1	4	2,34	1	5,3	0,32	?
U2	1,1	3	2	4,5	4,5	9,9	?
U3	4	4,4	4,5	0,3	7,4	2,3	?
U4	9,3	32	3	5	3,2	7.54	?
U5	-2	3	5.3	5,3	3,5	9,9	?
U6	-6,3	46,3	6,2	5	8,3	4,5	?
U7	3,5	5	3,2	5,3	6,2	7,8	?

Items

	I1	I2	I3	I4	I5	I6	I7
U1	1	?	?	?	?	?	3
U2	?	3	?	4,5	4,5	?	?
U3	4	?	4,5	?	?	?	4
U4	?	?	3	5	?	?	?
U5	?	3	?	?	3,5	?	?
U6	?	?	?	5	?	4,5	3
U7	3,5	5	?	5	?	?	3

Collaborative-based methods

Conceptual goal:

Give me recommendation based on a collaborative approach that leverages the ratings and actions of my peers and myself

Input:

User ratings + community ratings

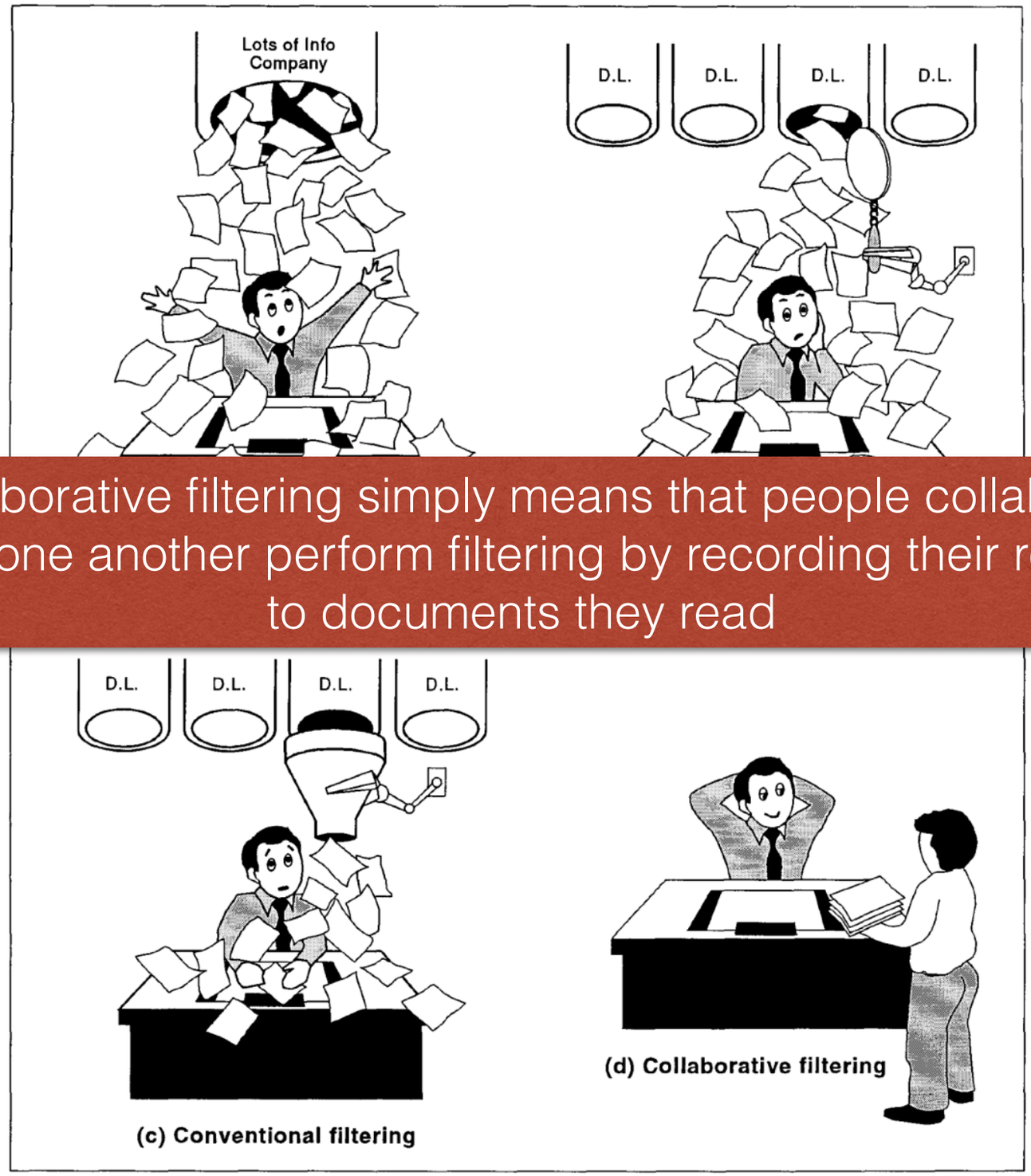
Using COLLABORATIVE FILTERING to Weave an Information TAPESTRY

David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry

Tapestry is an experimental mail system developed at the Xerox Palo Alto Research Center. The motivation for Tapestry comes from the increasing use of electronic mail, which is resulting in users being inundated by a huge stream of incoming documents [2, 7, 12]. One way to handle large volumes of mail is to provide mailing lists, enabling users to subscribe only to those lists of interest to them. However, as illustrated in Figure 1, the set of documents of interest to a particular user rarely map neatly to existing lists. A better solution is for a user to specify a *filter* that scans all lists, selecting interesting documents no matter what list they are in. Several mail systems support filtering based on a document's contents [3, 5, 6, 8]. A basic tenet of the Tapestry work is that more effective filtering can be done by involving humans in the filtering process.

Using collaborative filtering to weave an information tapestry
D Goldberg, D Nichols, BM Oki, D Terry
Communications of the ACM 35 (12), 61-70

3887 1992



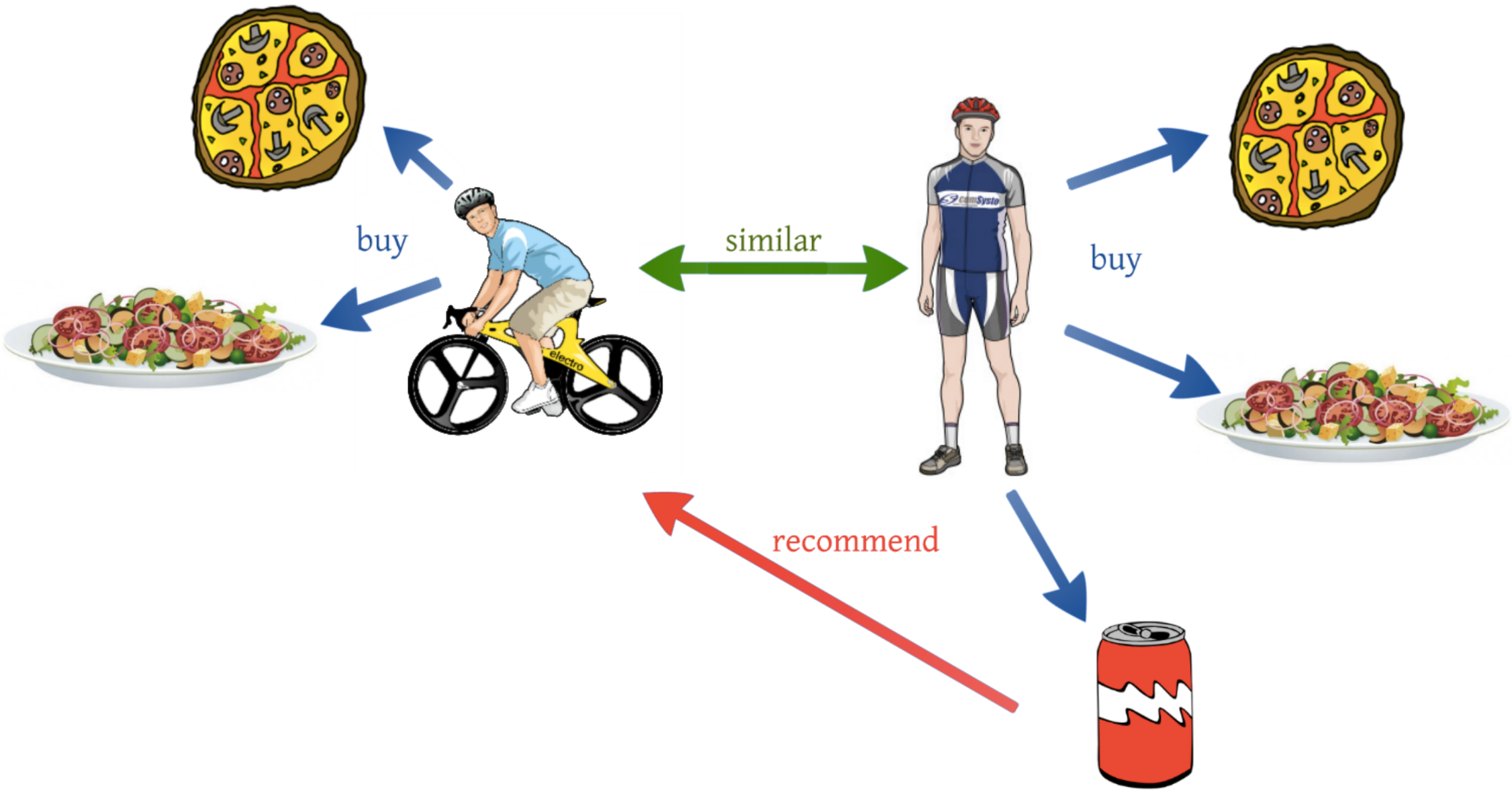
Collaborative filtering simply means that people collaborate to help one another perform filtering by recording their reactions to documents they read

Image from : http://dl.acm.org/citation.cfm?id=138867

Using collaborative filtering to weave an information tapestry
D Goldberg, D Nichols, BM Oki, D Terry
Communications of the ACM 35 (12), 61-70

3887 1992

Collaborative filtering



Collaborative filtering

- Collaborative filtering methods are based on collecting and **analyzing a large amount of information** on users' behaviors, activities or preferences and predicting what users will like based on their **similarity to other users**.

- Hypothesis: **Similar users tend to like similar items**.

- Requires a user community.

Rating Matrices

- *R* is the *m x n* rating matrix where *m* is the number of users and *n* the number of items.

- Rating can be defined in a variety of ways:

  - **Continuous ratings**: from -10 to 10

  - **Interval-based ratings**: 5 stars, 3 stars

  - **Ordinal ratings**: {strongly disagree, disagree, neutral, agree and strongly agree}

  - **Binary ratings**: Like/dislike

  - **Unary ratings**: Buy

Collaborative filtering

- **Cold Start**: There needs to be enough other users already in the system to find a match.

- **Sparsity**: If there are many items to be recommended, even if there are many users, the user/ratings matrix is sparse, and it is hard to find users that have rated the same items.

- **First Rater**: Cannot recommend an item that has not been previously rated.
  - New items
  - Esoteric items

- **Popularity Bias**: Cannot recommend items to someone with unique tastes.
  - Tends to recommend popular items.

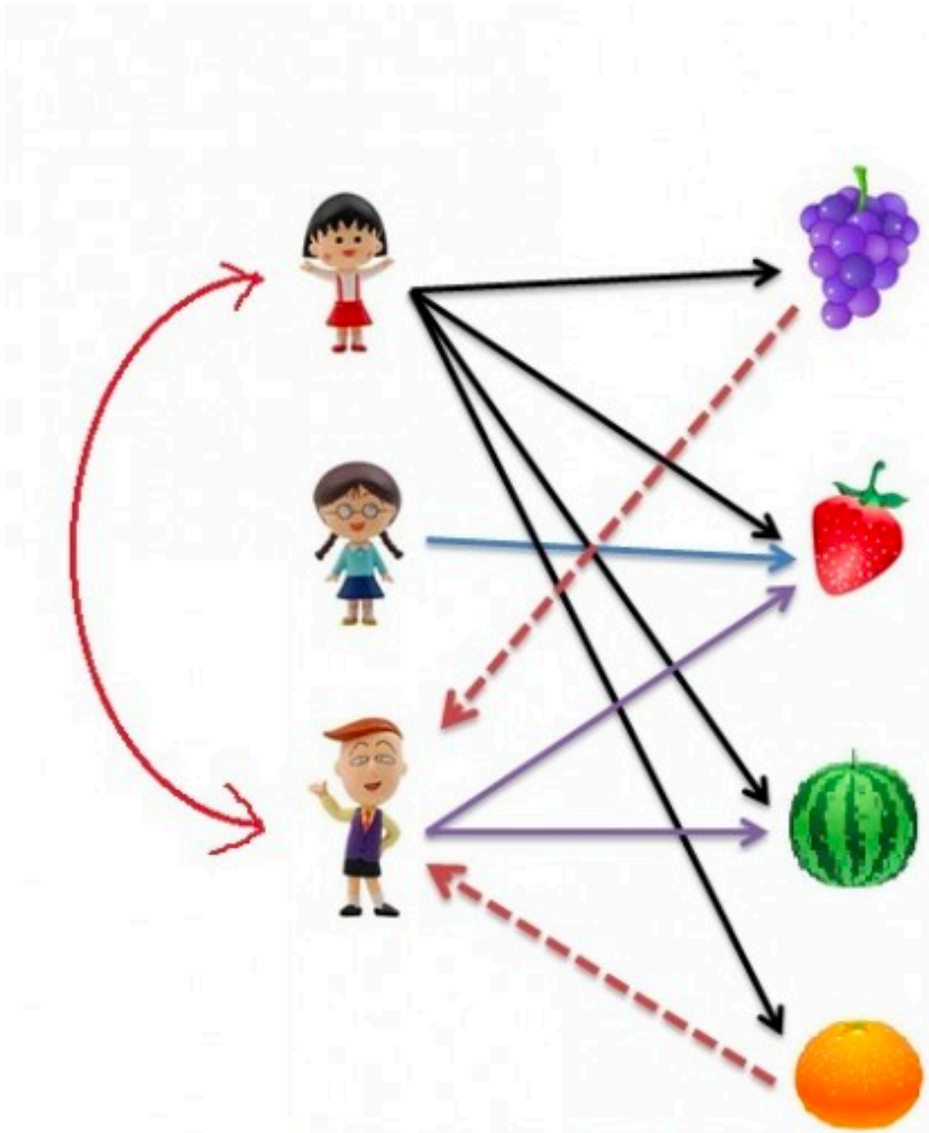
Two main approaches

1. Memory-based methods
    - Neighborhod-based methods
2. Model-based methods
    - Machine learning and data mining

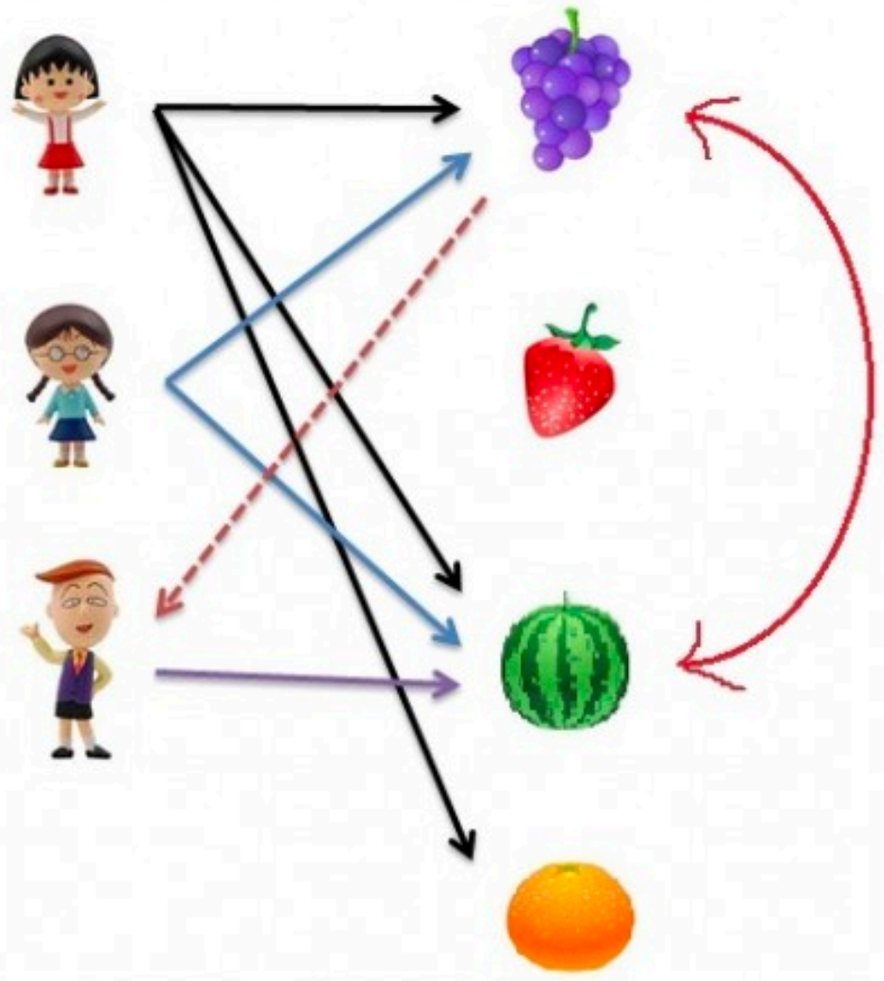
Neighborhood-based methods

- Neighborhood-based methods were among the earliest algorithms developed for collaborative filtering. These methods are based on the fact that similar users display similar patters or rating behaviors and similar items receive similar ratings.
- There are two primary types of neighborhood -based algorithms:
  - **User-based** CF works like this: take a user U and a set of other users D whose ratings are similar to the ratings of the selected user U. And, use the ratings from those like-minded users to calculate a prediction for the selected user U.
  - In **Item-based** CF you build an item-item matrix determining relationships between pairs of items and using this matrix and data on the current user, infer the user's taste.

Neighborhood-based methods



User-based filtering



Item-based filtering

Let's see how we can create a **User-Based CF** for Movie recommendations.

EXAMPLE: Movie Recommender System. User-Based Collaborative Filtering

- Given an "active **user**" and an **item** that has not been seen by the **user**, the goal is to estimate the rating for the **item**.

	Superman	Star Wars 1	Matrix	Spiderman
Santi	3	3.5	4.5	¿?
User1	3.5	4	5	5
User2	3	¿?	4.5	3
User3	3.5	5	3.5	2

The basic technique

- User-based nearest-neighbor collaborative filtering
  - Given an "active user" (Santi) and an item I not yet seen by Alice
  - The goal is to estimate Santi's rating for the unseen items, e.g., by
    - find a set of users (peers) who liked the same items as Santi in the past **and** who have rated them
    - use, e.g. the average of their ratings to predict, if Santi will like them
    - do this for all items Santi has not seen and recommend the best-rated

	Superman	Star Wars 1	Matrix	Spiderman
Santi	3	3.5	4.5	¿?
User1	3.5	4	5	5
User2	3	¿?	4.5	3
User3	3.5	5	3.5	2

How to measure similarity between users?

- The computation of the similarity between the items is one **critical step** in the CF algorithms.

- The basic idea in similarity computation between two users *a* and *b* is to first isolate the items commonly rated by both users (set *P*), and then to apply a similarity computation technique to determine the similarity.

Similarity Measures

- Euclidean Distance
- Pearson Correlation
- Person Correlation corrected
- Spearman Correlation
- Cosine Distance

How to measure similarity between users?

- Euclidean distance

$$sim(a,b) = \sqrt{\sum_{p \in P} (r_{a,p} - r_{b,p})^2}$$

- Pearson Correlation

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

- Cosine distance

$$sim(a,b) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

Where:

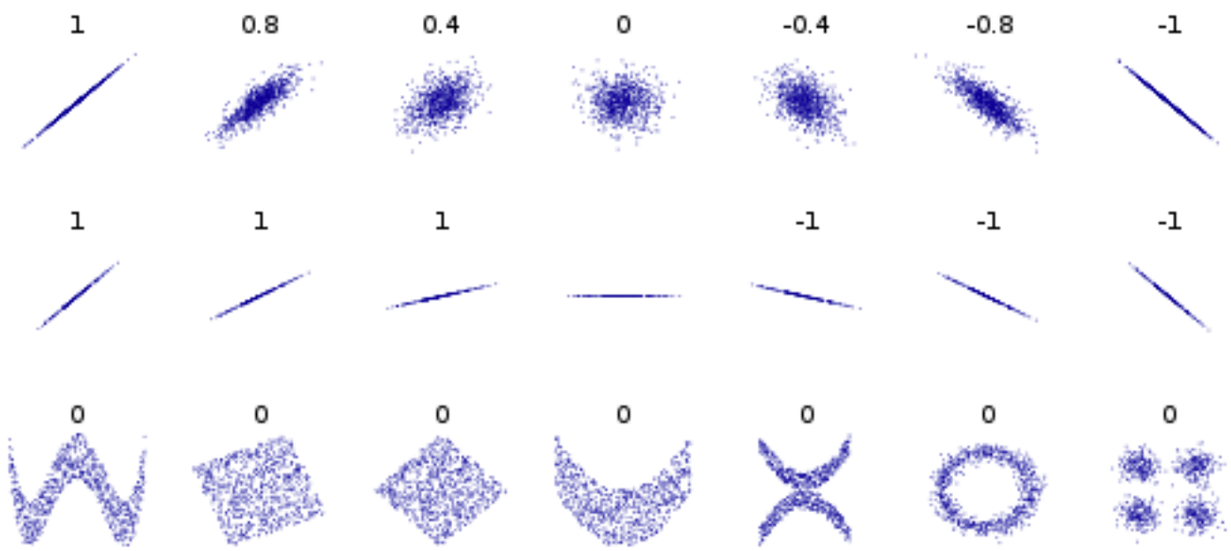
- $sim(a,b)$ is the similarity between user "a" and user "b"
- P is the set of common rated movies by user "a" and "b"
- $r_{a,p}$ is the rating of movie "p" by user "a"
- $\bar{r}_a$ is the mean rating given by user "a"

Similarity Measures
Euclidean distance

$$sim(u,v) = \sqrt{\sum_{j \in P} (r_{uj} - r_{vj})^2}$$

CAUTION: if the users use to rate with a different mean and standard deviation euclidean distance can give some problems

Similarity Measures
Pearson Correlation



Negative Values?

Strange correlations are rare,
and do not carries interesting information

Which is the best measure?

Hands on time!

Which is the best similarity function?

- …. there is not a clear answer…but, there are some tips:

  - Pearson Correlation used to work better than euclidean distance since it is based more on the ranking than on the values.

  - In general, Pearson Correlation coefficient is preferable to the raw cosine because of the bias adjustment effect of mean-centering

  - Cosine distance is usually used when our data is binary/unary, i.e. “like vs. not like” or “buy vs. not buy”.

Similarity Measures

- *Significance weighting*: When two users (or items) have **very few** items (or users) **in common** the **reliability** of the similarity scores if **low**. In this cases the similarity score should be reduced with a discount factor to de-emphasize the importance of that pair the importance of that user pair.

$$sim_c(u,v) = sim(u,v) \times \frac{1}{min(50, |I_u \cap I_v|)}$$

How do we generate a prediction from the neighbor's ratings?

$$\hat{r}_{u,j} = \frac{\sum_{v \in P_u(j)} sim(u,v) \times r_{v,j}}{\sum_{v \in P_u(j)} sim(u,v)}$$

Where *Pu(j)* is denoted to the set of the top-k similar users to target user *u,* sim(u,v) the similarity between user u and v and $r_{u,j}$ the rating of the user *u* to the movie *j*

How do we generate a prediction from the neighbor's ratings?

Example:

Critic	$sim(a,b)$	Rating Movie1: r_{b,p_1}	$sim(a,b) * (r_{b,p_1})$	Rating Movie2: r_{b,p_2}	$sim(a,b) * (r_{b,p_2})$
User1	0.99	3	2.97	2.5	2.48
User2	0.38	3	1.14	3	1.14
User3	0.89	4.5	4.0	-	-
User4	0.92	3	2.77	3	2.77
$\sum_{b \in N} sim(a,b) * (r_{b,p})$			10.87		6.39
$\sum_{b \in N} sim(a,b)$			3.18		2.29
$pred(a,p)$			3.41		2.79

Other prediction functions

- Different users may provide ratings on different scales. Some users rate all items highly, whereas another rate all items negatively

$$\hat{r}_{u,j} = \bar{r}_u + \frac{\sum_{v \in P_u(j)} sim(u,v) \times (r_{v,j} - \bar{r}_v)}{\sum_{v \in P_u(j)} sim(u,v)}$$

Caution: predicted scores can goes outside the range. However, the rank is correct.

Other prediction functions

$$\hat{r}_{u,j} = \bar{r}_u + \sigma_u \frac{\sum_{v \in P_u(j)} sim(u,v) \times z_{v,j}}{\sum_{v \in P_u(j)} |sim(u,v)|}$$

Where z_{vj} is the standardized rating computed as follows:

$$z_{uj} = \frac{r_{uj} - \bar{r}_u}{\sigma_u} = \frac{s_{uj}}{\sigma_u}$$

and $$\sigma_u = \sqrt{\frac{\sum_{j \in I_u} (r_{uj} - \bar{r}_u)^2}{|I_u| - 1}}$$

Caution: predicted scores can goes outside the range.
However, the rank is correct.

Some tricks (I)

- **Top-k** most similar users to the target user in order to do the predictions.

- Weakly correlated users might add to the error in prediction, as well as, negative correlations often do not have a predictive value.

Some tricks (II)

Recursive methods: In order to avoid cold-start we can apply a recursive method for new users or for sparse data sets.

Some tricks (III)

Similarlity amplificaton:

$$sim(u,v) = Pearson(u,v)^{\alpha}$$

Where $\alpha > 1$

Some tricks (IV)

Clustering applied as a “first step” for **shrinking** the candidate set of users.

Long Tail Problem

- Usually, popular items provide less profit that non-popular.

- Difficult to provide good rating prediction for those item in the long tail.

- Popular items does not provide the same information about tastes than non-popular items

Impact of the long Tail

Some movies are really popular since other very unpopular. Popular ratings can sometimes worsen the quality of the recommendations since they tend to be less discriminative across different users

Each item *j* can be weighted by w_j as follows

$$w_j = log\left(\frac{m}{m_j}\right)$$

where *m* is the total number of users, and m_j is the number of users who have rated the item j

$$Pearson(u,v) = \frac{\sum_{k \in I_u \cap I_v} w_k (r_{uk} - \bar{r}_u)(r_{vk} - \bar{r}_v)}{\sqrt{\sum_{k \in I_u \cap I_v} w_k (r_{uk} - \bar{r}_u)^2} \sqrt{\sum_{k \in I_u \cap I_v} w_k (r_{vk} - \bar{r}_v)^2}}$$

Clustering

- Computing the similarity between users is computationally expensive.
- Clustering is cheaper to than computing the mxm similarity matrix
- Redefine top similar users using only the subset of users in the same cluster
- Problem: MxM is an incomplete, really sparse, matrix.

Dimensionality Reduction and Neighborhood Methods

- Dimensionality reduction can **improve** neighborhood methods in terms of **accuracy** and also in terms of **efficiency**.

- Similarities are hard to be computed in huge dimensional sparse rating matrices.

  - Latent factor models

Sparse Linear Models

- Computes the item-item relations, by estimating an item x item sparse aggregation coefficient matrix **S**.

- The recommendation score of an unrated item **i** for a user **u** is:

$$\hat{r}_{ui} = \mathbf{r}_u^T \mathbf{s}_i.$$

$$\begin{aligned} \underset{S}{\text{minimize}} \quad & \tfrac{1}{2}\sum_{u,i}(r_{ui} - \hat{r}_{ui})^2 + \tfrac{\beta}{2}||S||_F^2 + \lambda||S||_1, \\ \text{subject to} \quad & S \geq 0, \text{and} \\ & \text{diag}(S) = 0. \end{aligned}$$

SLIM: Sparse linear methods for top-n recommender systems
X Ning, G Karypis
Data Mining (ICDM), 2011 IEEE 11th International Conference on, 497-506

115 2011

Explaining recommendations

- Help on:
  - Transparency
  - Trust

A survey of explanations in recommender systems N Tintarev, J Masthoff Data Engineering Workshop, 2007 IEEE 23rd International Conference on, 801-810	226	2007

What is an Explanation

- Additional data to help users understand a specific recommendation
    - It is totally separate from an explanation if how the system works as a whole
    - Some explanation are confidence values
- Show distributions
- Sometimes tied to ability to edit profile to improve recommendations..

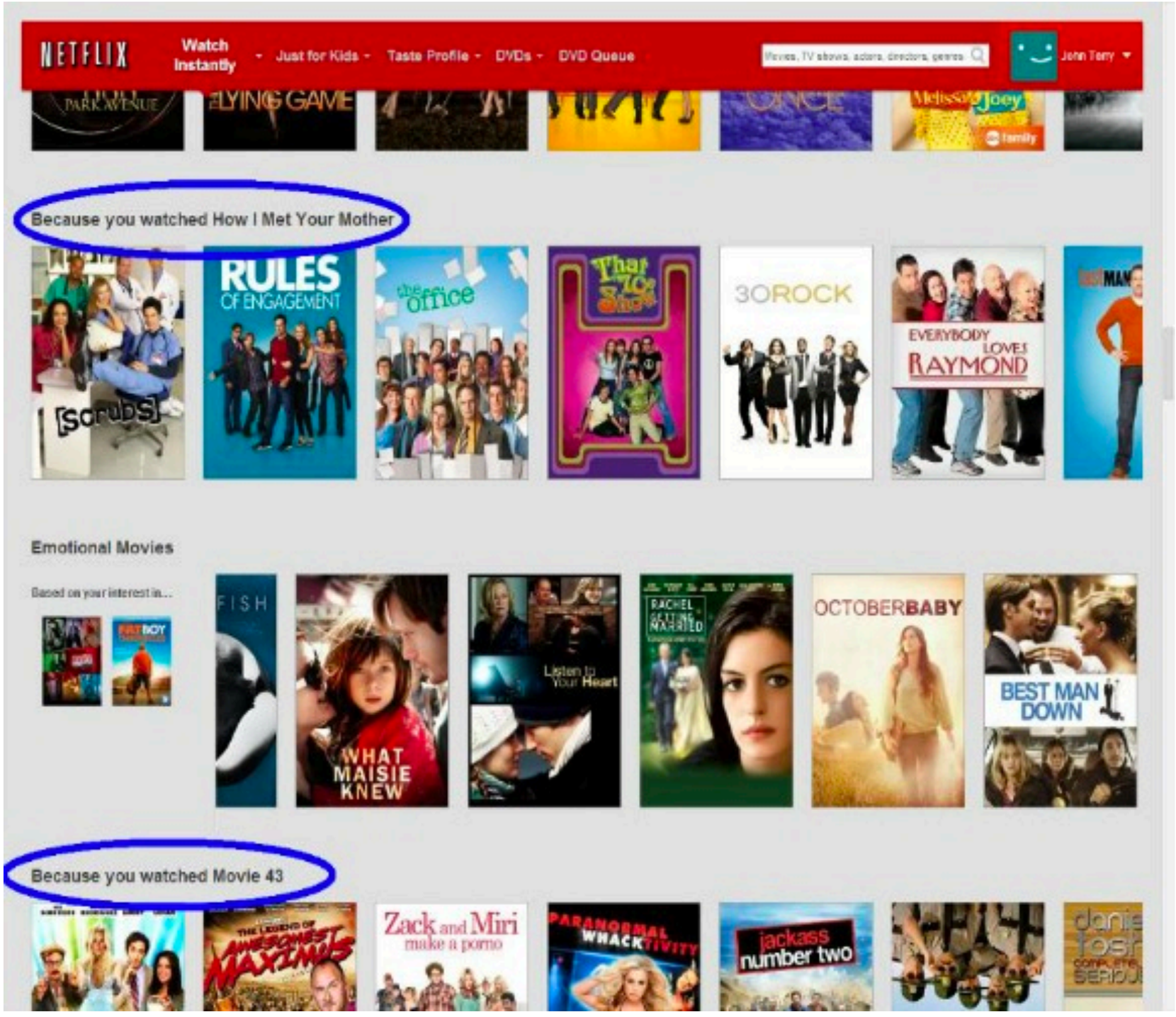
#		N	Mean Response	Std Dev
1	Histogram with grouping	76	5.25	1.29
2	Past performance	77	5.19	1.16
3	Neighbor ratings histogram	78	5.09	1.22
4	Table of neighbors ratings	78	4.97	1.29
5	Similarity to other movies rated	77	4.97	1.50
6	Favorite actor or actress	76	4.92	1.73
7	MovieLens percent confidence in prediction	77	4.71	1.02
8	Won awards	76	4.67	1.49
9	Detailed process description	77	4.64	1.40
10	# neighbors	75	4.60	1.29
11	No extra data – focus on system	75	4.53	1.20
12	No extra data – focus on users	78	4.51	1.35
13	MovieLens confidence in prediction	77	4.51	1.20
14	Good profile	77	4.45	1.53
15	Overall percent rated 4+	75	4.37	1.26
16	Complex graph: count, ratings, similarity	74	4.36	1.47
17	Recommended by movie critics	76	4.21	1.47
18	Rating and %agreement of closest neighbor	77	4.21	1.20
19	# neighbors with std. deviation	78	4.19	1.45
20	# neighbors with avg correlation	76	4.08	1.46
21	Overall average rating	77	3.94	1.22

Table 1. Mean response of users to each explanation interface, based on a scale of one to seven. Explanations 11 and 12 represent the base case of no additional information. Shaded rows indicate explanations with a mean response significantly different from the base cases (two-tailed $\alpha = 0.05$).

Figure 1. One of the twenty-one different explanation interfaces given shown in the user survey. Notice that the title has been encoded, so that it does not influence a user's decision to try a movie.

Explaining collaborative filtering recommendations
JL Herlocker, JA Konstan, J Riedl
Proceedings of the 2000 ACM conference on Computer supported cooperative ... 1264 2000

Example


NETFLIX
Watch Instantly
Just for Kids
Taste Profile
DVDs
DVD Queue
John Terry
Because you watched How I Met Your Mother
RULES OF ENGAGEMENT
the office
That '70s Show
30 ROCK
EVERYBODY LOVES RAYMOND
Emotional Movies
Based on your interest in...
FISH
WHAT MAISIE KNEW
Listen to Your Heart
OCTOBERBABY
BEST MAN DOWN
Because you watched Movie 43
Zack and Miri make a porno
PARANORMAL WHACKTIVITY
jackass number two

ACM RecSys
Challenge

ACM RecSys Challenge 2017

ACM RecSys Challenge 2017

About ▾ Participate ▾ Timeline Workshop ▾ Challenge 2016 RecSys conference

We are currently finishing the datasets, evaluation metrics and submission system. Please follow us on Twitter to get updates: @recsyschallenge

About

The ACM RecSys Challenge 2017 is focussing on the problem of job recommendations on XING in a cold-start scenario. The challenge will consists of two phases:

1. **offline evaluation:** fixed historic dataset and fixed targets for which recommendations/solutions need to be computed/submitted.
2. **online evaluation:** dynamically changing targets. Recommendations submitted by the teams will actually be rolled out in XING's live system (details about the online evaluation such as the approach for ensuring fair conditions between the teams will follow).

Both phases aim at the following task:

> **Task:** given a new job posting p, the goal is to identify those users that (a) may be interested in receiving the job posting as a push recommendation and (b) that are also appropriate candidates for the given job.

For both offline and online evaluation, the same evaluation metrics and the same types of data sets will be used. The offline evaluation is essentially used as an entry gate to the online evaluation:

http://2017.recsyschallenge.com/

job recommendations on XING in a **cold-start scenario**.

http://2017.recsyschallenge.com/

Task nº1

Read the full documentation and think about the problem
be ready for a debate!

Timeline ▲top

When?	What?
Beginning of March	**RecSys challenge starts:** • RecSys challenge starts with *offline evaluation* • Submission system will be available via recsys.xing.com (currently offline) • Maximum number of submissions per day: 20
April 16th (23:59 Hawaiian time)	**Offline evaluation ends:** • Based on the overall leaderboard of the *offline challenge*, the top teams (probably top 20) are asked to join the *online challenge* • Top teams get access to API of the online challenge. • The submission of the offline challenge will continue to be open • After this deadline there will be a possibility to join the online challenge for teams who achieve awesome scores
May 1st	**Online challenge starts:** • Every day, new target items will be released for which the teams are supposed to compute recommendations, i.e. identify users that may be interested in these items. • Teams download the new target list via API, compute recommendations and submit their solutions via API. Teams can script their recommender systems to regulalry pull from the API to check for updates.
June 4th (23:59 Hawaiian time)	**Online evaluation ends:** • Last day of online evaluation • Submission system closes
June 12th	• Official results will be announced • Winner of the challenge = winner of the online challenge