Recommender systems

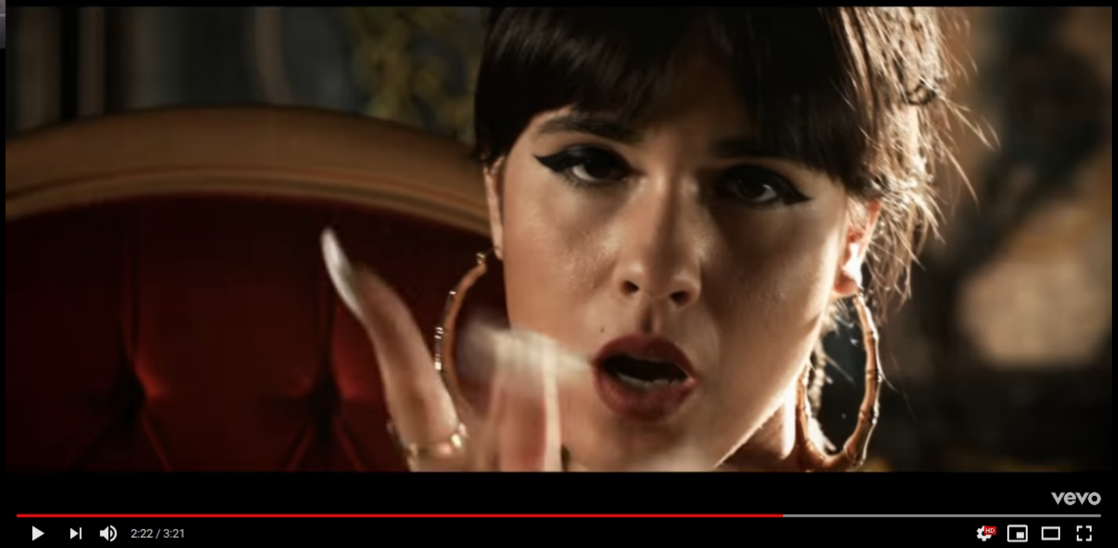
Mining Massive Datasets
Carlos Castillo
Topic 08

Universitat Pompeu Fabra Barcelona

Sources

- Data Mining, The Textbook (2015) by Charu Aggarwal (Section 18.5) – slides by Lijun Zhang
- Mining of Massive Datasets 2nd edition (2014) by Leskovec et al. (Chapter 9) - slides A, B

YouTube's algorithm cares as much about trap as your professor, but manages to produce reasonable recommendations. How?



Recommender systems (purchase)

- Given data from user buying behaviors
  - User profiles, interests, browsing behavior, buying behavior, and ratings about various items
- Leverage such data to make recommendations to customers about possible buying interests

Recommender systems (general)

- Given data from user interests
  - User profiles, interests, browsing behavior, item interaction behavior, ratings about various items
- Leverage such data to make recommendations to users about further interesting items

NETFLIX

Spotify®

amazon

YouTube

STEAM®

Utility matrix

- For n users and d items, there is a matrix D of **utility** values
  - The utility value for a user-item pair could correspond, e.g., to buying behavior or ratings of the user for the item
  - Typically, a small subset of the utility values are known

Utility matrix (ratings-based, positive preference)

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			5		2
U_2		5			4	
U_3	5	3		1		
U_4			3			4
U_5				3	5	
U_6	5		4			

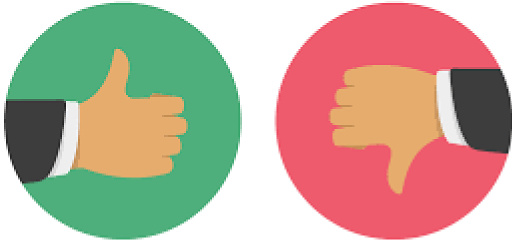
(a) Ratings-based utility

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			1		1
U_2		1			1	
U_3	1	1		1		
U_4			1			1
U_5				1	1	
U_6	1		1			

(b) Positive-preference utility

Types of utility

- **Explicit:** we ask users to rate items



- **Implicit:** we take watching/consuming/buying behavior as a positive signal, skip/hide as negative

Sources for a recommendation

- Content-based recommendation
  - Users and items are associated with features
  - Features are matched to infer interest
- Interaction-based recommendations
  - Leverage user preferences in the form of ratings or other behavior
  - Recommend through similarity or latent factors

THE COLD START PROBLEM

New items have no ratings

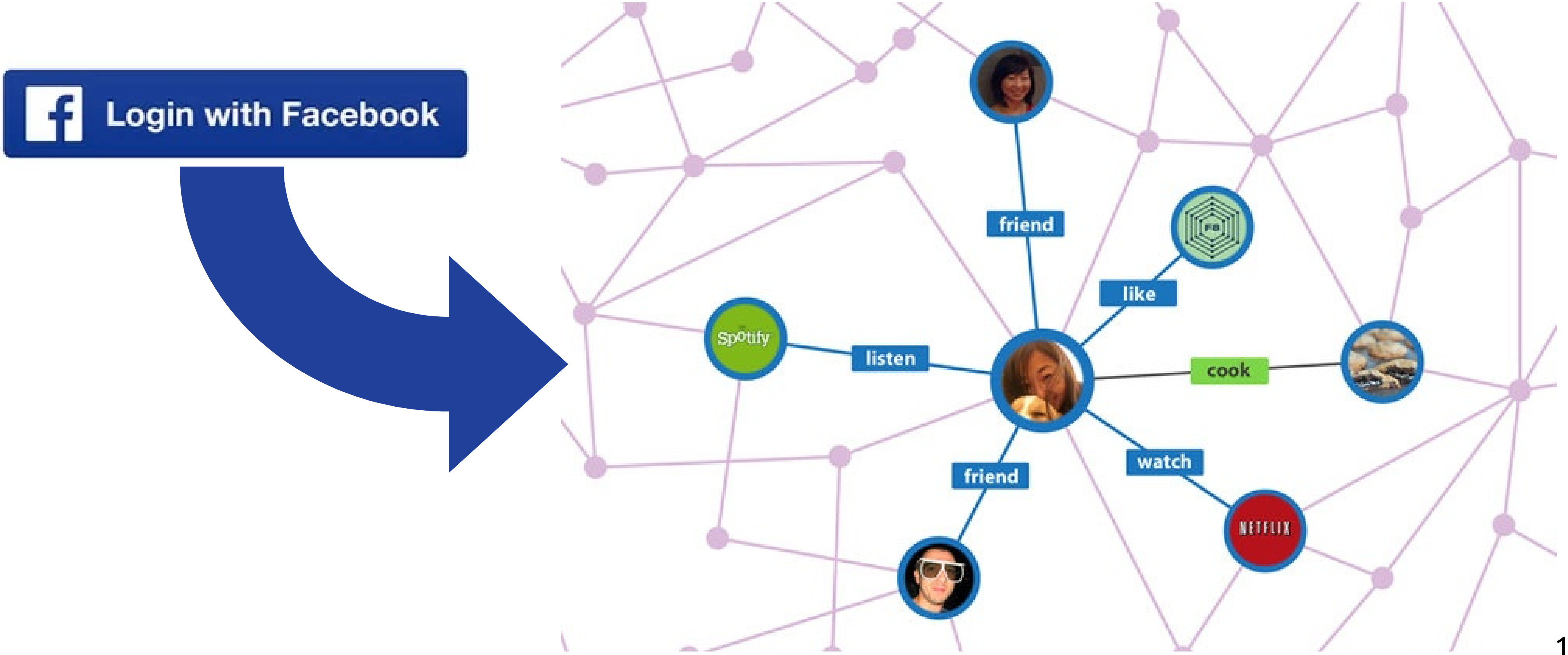
and

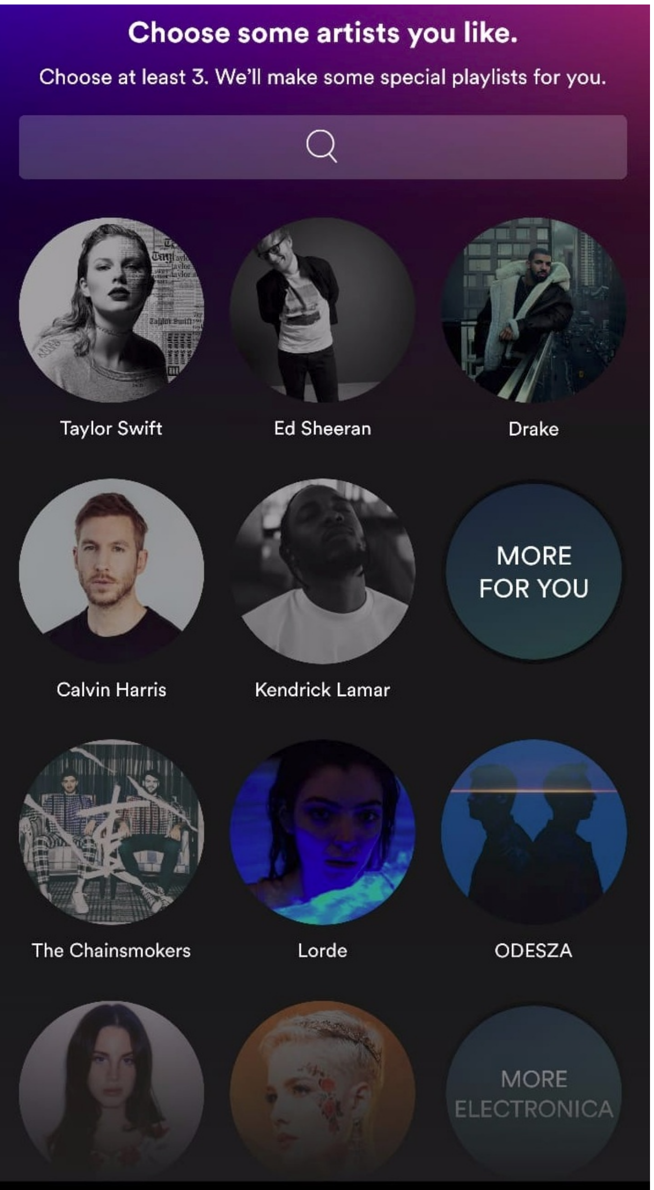
New users have no history

Photo: Torque News

THE COLD START PROBLEM

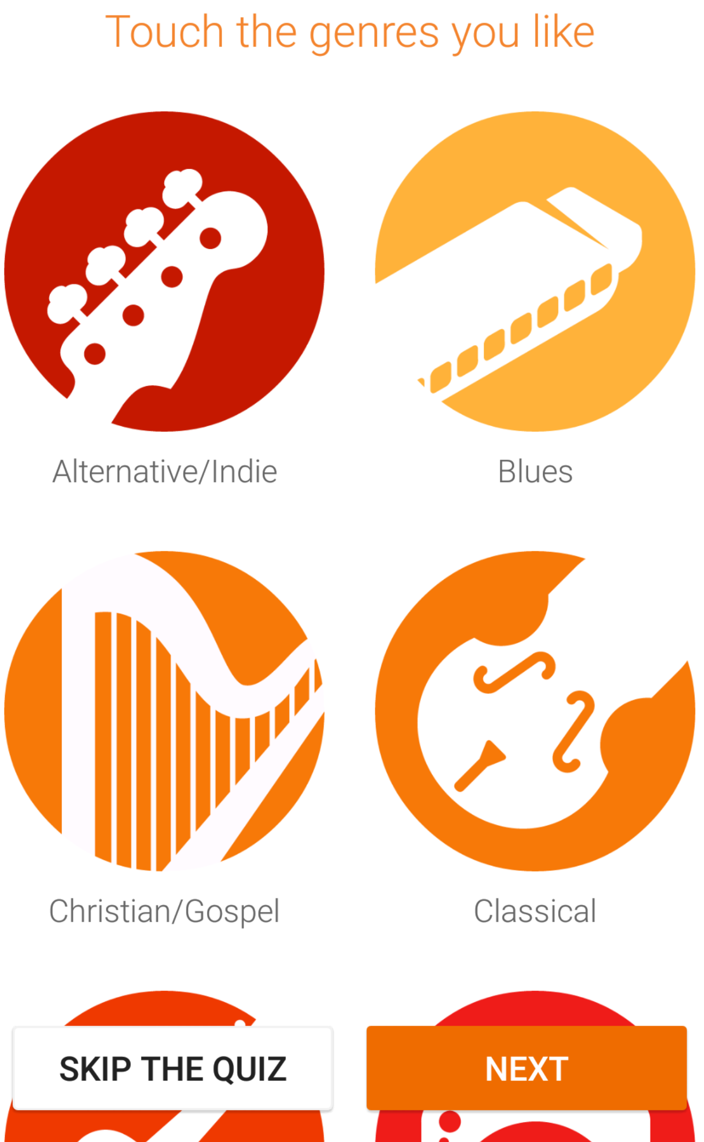
Solution 1. “Side information”





THE COLD START PROBLEM

Solution 2. “On-boarding” users



Content-based recommendations

General idea of content-based recommendations

- Movies: recommend other movies with same director, actor, genre, as viewed ones
- Products: recommend other products in same category, brand, color, as purchased ones

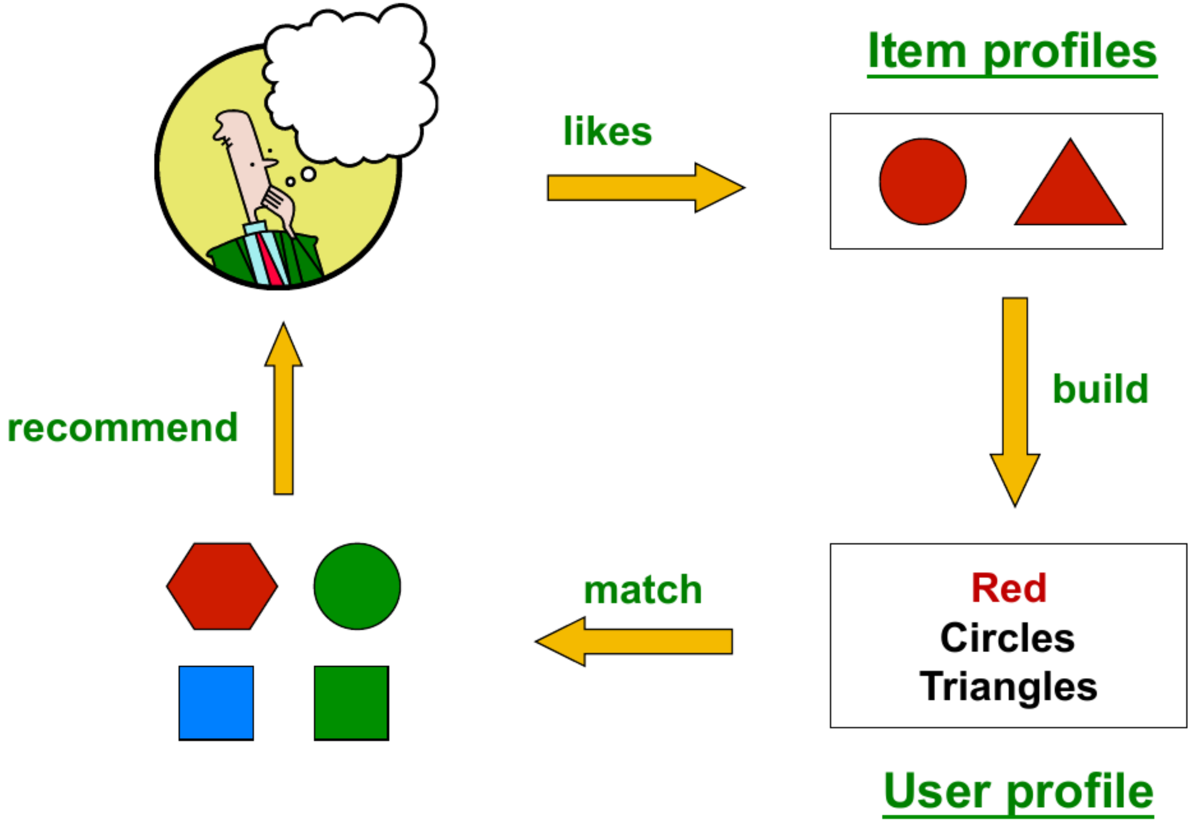
Creating a recommendation

- User is associated with some documents that describe his/her interests
  - Specified demographic profile
  - Specified interests at registration time
  - Descriptions of the items bought
- Items are also associated with semi-structured descriptions

JBL GO lleva el sonido de calidad JBL a todas partes. GO es su solución de altavoz todo en uno y reproduce música en tiempo real vía Bluetooth desde smartphones y tabletas, gracias a su batería recargable. También cuenta con un práctico manos libres.

Potencia	**3 W**
Respuesta de Frecuencia	**180Hz – 20 kHz**
Tipo de altavoz	**Portátil**
Amplificador de sonido	**Integrado**

Creating a recommendation (cont.)



Mining of Massive Datasets 2nd edition (2014) by Leskovec et al. (Chapter 9) - slides A, B

Possible recommendation methods

- **If no utility matrix is available**
  - k-nearest neighbor approach
    - Find the top-k items that are closest to the user (when items and users can be represented in the same space, e.g., dating apps)
  - The cosine similarity with tf-idf can be used
- **If a utility matrix is available**
  - Classification-based approach: training documents are those for which the user has specified utility, labels are utility values
  - Regression-based approach in the case of ratings
- Limitations: depends on the quality of the features

Example: regression-based approach for content-based recommendation

Movie	Adventure	Action	Science-Fiction	Drama	Crime	Thiller		User 1	User 2
Star Wars IV	1	1	1	0	0	0		1	-1
Saving Private Ryan	0	0	0	1	0	0			
American Beauty	0	0	0	1	0	0			
City of Gold	0	0	0	1	1	0		-1	1
Interstellar	0	0	1	1	0	0		1	
The Matrix	1	1	1	0	0	1			1

...

We would do two regressions: one for the ratings of user 1 and another for user 2.

How many rated movies would we need, as a minimum, to be able to do this?

Pros and Cons of content-based recommendations

- Pros:
  - No cold-start problem if no utility needed
  - Able to recommend to users with very particular tastes
  - Able to recommend new and obscure items
  - Able to provide explanations that are easily understandable

Pros and Cons of content-based recommendations

- Cons:
  - Finding the correct features might be hard
  - Recommending for new users still challenging if user features are different from item features
  - Overspecialization/"bubble": might reinforce user interests
  - Does not exploit ratings of other users!

Interaction-based recommendations

Missing-value estimation/completion

- The matrix is extremely large and sparse

$$M = \begin{bmatrix} \blacksquare & & \blacksquare & & & \blacksquare & \\ & \blacksquare & & \blacksquare & \blacksquare & & \blacksquare \\ \blacksquare & & & \blacksquare & & \blacksquare & \\ & & \blacksquare & & \blacksquare & & \\ & \blacksquare & & \blacksquare & & & \blacksquare \end{bmatrix} \in \mathbb{R}^{n \times d}$$

Only black squares have non-zero values.

Types of algorithms

- Neighborhood-Based Methods
  - User-Based or Item-Based Similarity with Ratings
- Graph-Based Methods
- Clustering Methods
  - Adapting k-Means Clustering or Adapting Co-Clustering
- Latent Factor Models
  - Matrix Factorization, e.g., Singular Value Decomposition

User-based similarity with ratings

- Let $I_{u,v}$ be common ratings between two users
- Similarity: Pearson correlation coefficient

$$\mathrm{sim}(u,v) = \frac{\sum_{i \in I_{u,v}} (u_i - \hat{u}) \cdot (v_i - \hat{v})}{\sqrt{\sum_{i \in I_{u,v}} (u_i - \hat{u})^2 \cdot \sum_{i \in I_{u,v}} (v_i - \hat{v})^2}}$$

$$\hat{u} = \frac{1}{|u|} \sum_{i=1}^{|u|} u_i \qquad \hat{v} = \frac{1}{|v|} \sum_{i=1}^{|v|} v_i$$

Note: averages are taken over all elements, not only ones in common

User-based similarity with ratings (cont.)

$$\text{sim}(u,v) = \frac{\sum_{i \in I_{u,v}} (u_i - \hat{u}) \cdot (v_i - \hat{v})}{\sqrt{\sum_{i \in I_{u,v}} (u_i - \hat{u})^2 \cdot \sum_{i \in I_{u,v}} (v_i - \hat{v})^2}}$$

- Score of recommendation

$$\text{score}(u,i) = \hat{u} + \frac{\sum_{v: v_i \neq \text{NULL}} \text{sim}(v,u) \cdot (v_i - \hat{v})}{\sum_{v: I_{u,v} \neq \emptyset} |\,\text{sim}(v,u)|}$$

Note: for efficiency one can take only the most similar users

Try it!

	2			4	5	
	5		4			1
			5		2	
		1		5		4
u			4			2
	4	5		1		

1. Compute avg(v) for all users
2. Compute sim(u,v) for all users for which there is some intersection with u
3. Compute score(u,i) for all items that u has not seen yet

$$\mathrm{sim}(u,v) = \frac{\sum_{i \in I_{u,v}} (u_i - \hat{u}) \cdot (v_i - \hat{v})}{\sqrt{\sum_{i \in I_{u,v}} (u_i - \hat{u})^2 \cdot \sum_{i \in I_{u,v}} (v_i - \hat{v})^2}}$$

$$\mathrm{score}(u,i) = \hat{u} + \frac{\sum_{v: v_i \neq \mathrm{NULL}} \mathrm{sim}(v,u) \cdot (v_i - \hat{v})}{\sum_{v: I_{u,v} \neq \emptyset} |\mathrm{sim}(v,u)|}$$

Exercise from ML for Recommender Systems tutorial by Alex Karatzoglou, 2015.

You can do the same with items!

- Item-based similarities with ratings

$$\text{sim}(i,j) = \frac{\sum_{u \in I_{i,j}} (u_i - \hat{i}) \cdot (u_j - \hat{j})}{\sqrt{\sum_{u \in I_{i,j}} (u_i - \hat{i})^2 \cdot \sum_{u \in I_{i,j}} (u_j - \hat{j})^2}}$$

- Item-based recommendations

$$\text{score}(u,i) = \hat{u} + \frac{\sum_{j : u_j \neq \text{NULL}} \text{sim}(i,j) \cdot (u_j - \hat{j})}{\sum_{j : I_{i,j} \neq \emptyset} |\,\text{sim}(i,j)|}$$

Try it!

2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

1. Compute avg(j) for all items
2. Compute sim(i,j) for all items for which there is some intersection with i
3. Compute score(u,i) for all users who have not seen i yet

$$\mathrm{sim}(i,j) = \frac{\sum_{u \in I_{i,j}} (u_i - \hat{i}) \cdot (u_j - \hat{j})}{\sqrt{\sum_{u \in I_{i,j}} (u_i - \hat{i})^2 \cdot \sum_{u \in I_{i,j}} (u_j - \hat{j})^2}}$$

$$\mathrm{score}(u,i) = \hat{u} + \frac{\sum_{j : u_j \neq \mathrm{NULL}} \mathrm{sim}(i,j) \cdot (u_j - \hat{j})}{\sum_{j : I_{i,j} \neq \emptyset} |\mathrm{sim}(i,j)|}$$

Exercise from ML for Recommender Systems tutorial by Alex Karatzoglou, 2015.

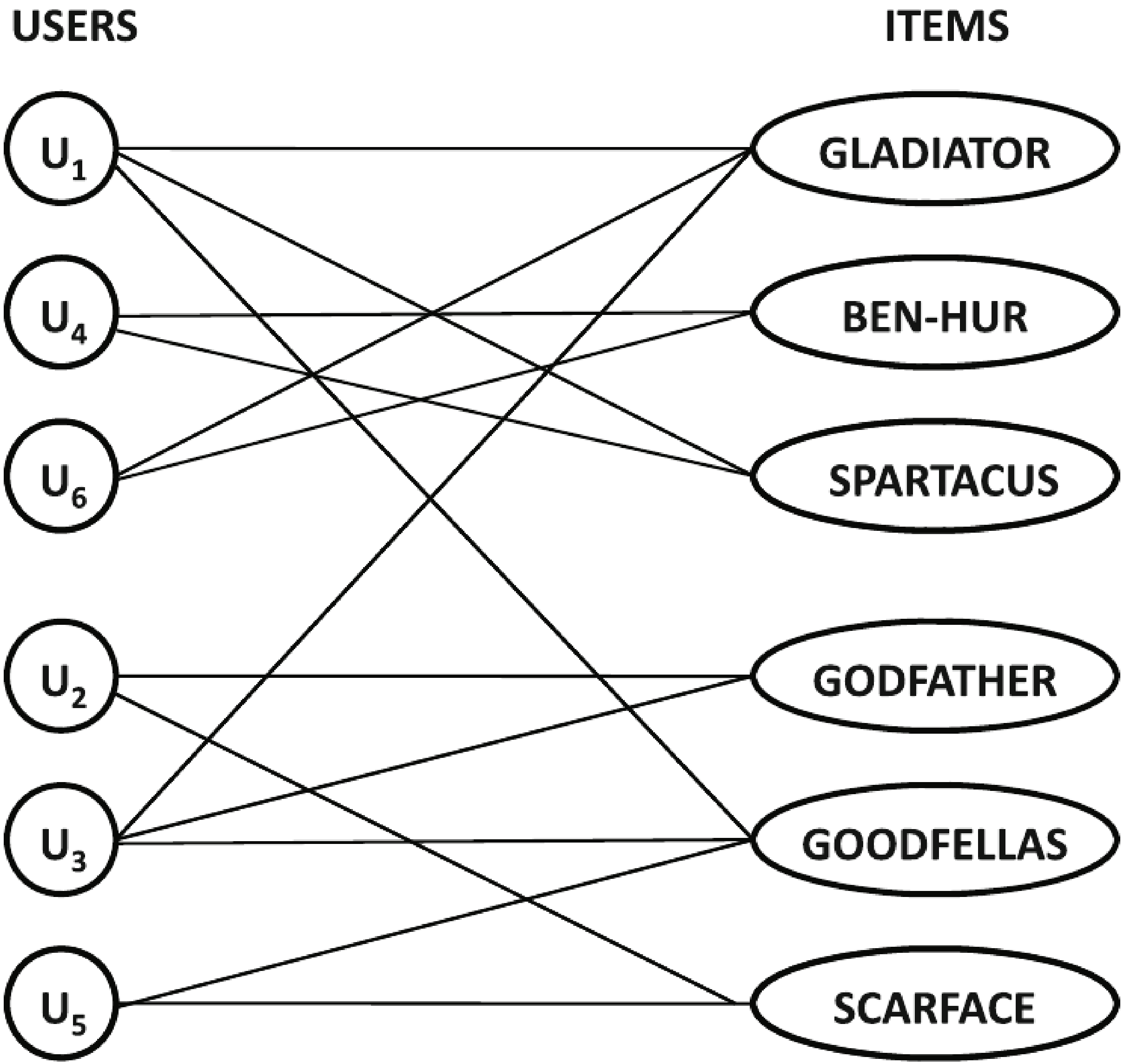
Note

- There are many ways of computing user-based similarity and item-based similarity
- There are many ways of using these to generate recommendations
- The method we have described is aware of the **bias of users**, in the sense of some users being more positive/negative than others in general

Graph- and clustering-based methods

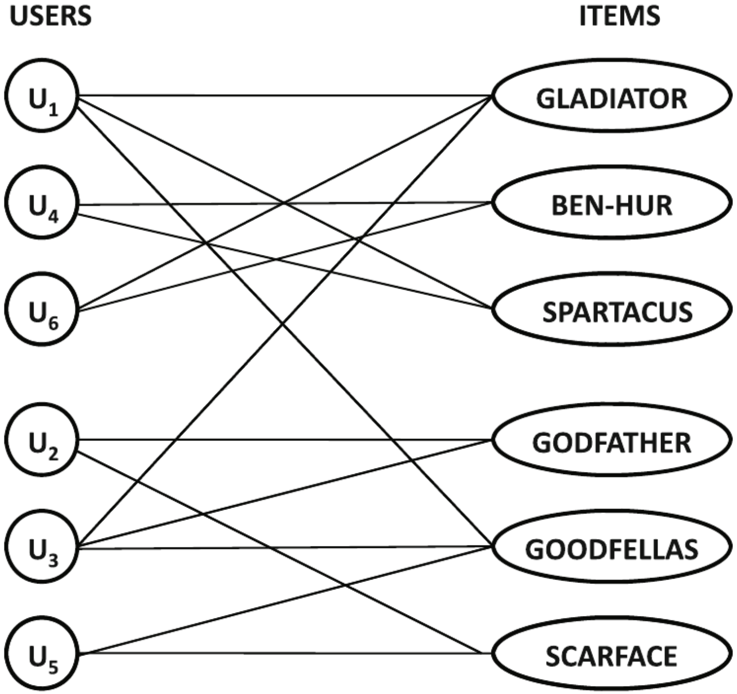
Graph-based methods

- Bipartite user-item graph with nodes N_u U N_i
- N_u users
- N_u items
- Non-zero utility ⇒ edge



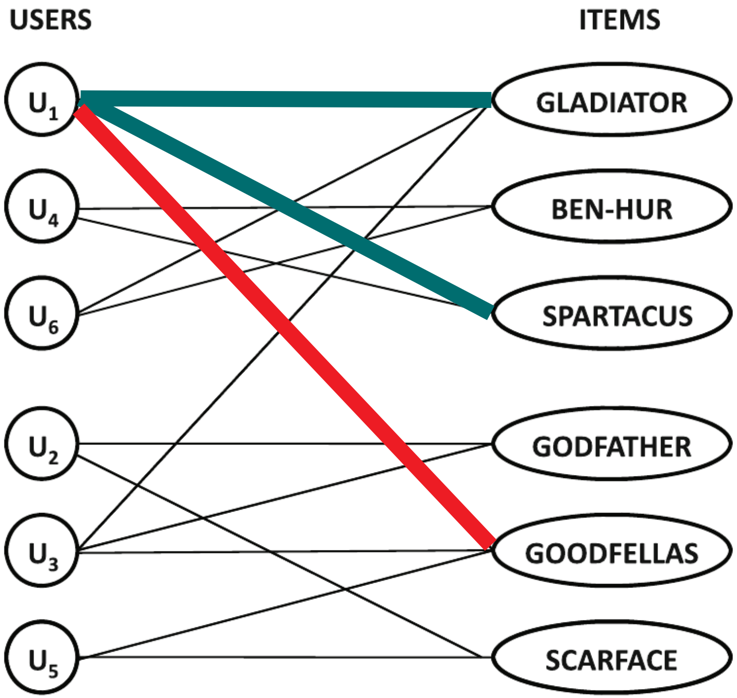
Graph-based methods (cont.)

- Use graph-based methods
  - Random walk with restart to a user or item
  - SimRank
- Low “random jump” probability might favor popular items



Graph-based methods (cont.)

- Signed networks can be used
  - Remember we must interpret ratings with respect to user and item averages
  - Below average rating ⇒ -
  - Above average rating ⇒ +
- Positive link prediction problem



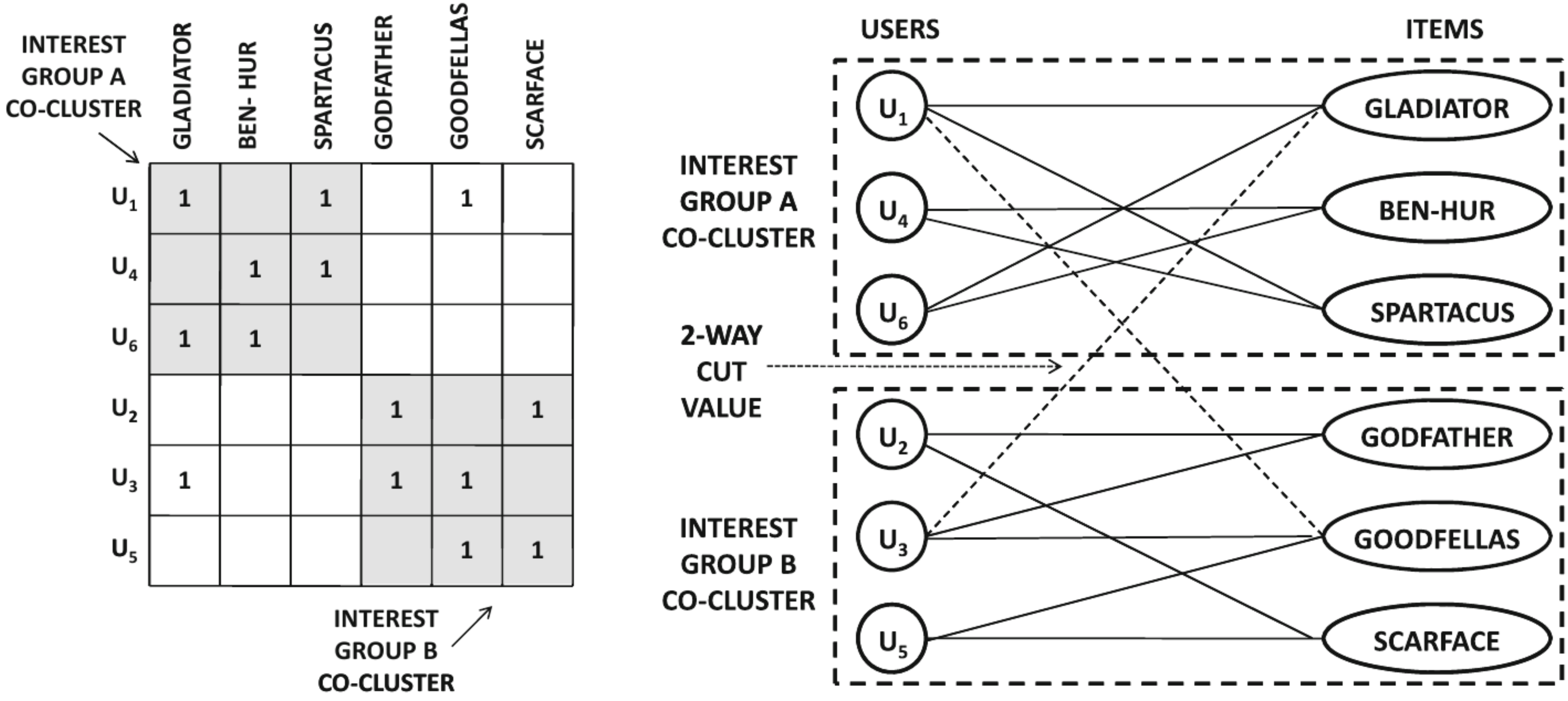
Clustering methods

- Motivations
  - Reduce computational cost
  - To some extent address data sparsity
- Results of clustering
  - Clusters of users for user-user similarity recs.
  - Clusters of items for item-item similarity recs.

Clustering methods (cont.)

- User-user recommendation approach
  - Cluster users into groups
  - For any user u, compute average normalized rating for each item i the user has not seen
  - Report these ratings for (u,i)
- Same with item-item recommendations
- Neighborhoods will be smaller

Co-Clustering Approach



(a) Co-cluster

(b) User-item graph

Latent factor models

Key idea

- Summarize the correlations across rows and columns in the form of lower dimensional vectors, or **latent** factors
- These latent factors become **hidden** variables that encode the correlations in the data matrix in a concise way and can be used to make **predictions**
- Estimation of the k-dimensional dominant latent factors is often possible even from **incompletely** specified data

Modeling

- n users: $\overline{U_1}, \ldots, U_n \in \mathbb{R}^k$
- d items: $\overline{I_1}, \ldots, I_d \in \mathbb{R}^k$
- Approximate rating r_{ij} by

$$r_{ij} \approx \left\langle \overline{U_i}, \overline{I_j} \right\rangle = \overline{U_i}^T \overline{I_j} = \overline{I_j}^T \overline{U_i}$$

- Approximate rating matrix $D = [r_{ij}]_{n \times d}$

$$D \approx F_{\text{user}} F_{\text{item}}^T \qquad F_{\text{user}} \in \mathbb{R}^{n \times k}$$

$$F_{\text{item}} \in \mathbb{R}^{d \times k}$$

Singular Value Decomposition

- SVD $D = Q\Sigma P^T$ $Q^TQ = I, P^TP = I$

  $D \in \mathbb{R}^{n\times d}$

  $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_d) \in \mathbb{R}^{d\times d}, \sigma_1 \geq \cdots \geq \sigma_d$

- Truncated SVD $D \approx Q_k \Sigma_k P_k^T$

  $\Sigma_k = \mathrm{diag}(\sigma_1, \ldots, \sigma_k) \in \mathbb{R}^{k\times k}, \sigma_1 \geq \cdots \geq \sigma_k$

- Note: SVD is undefined for incomplete matrices

Matrix factorization

- SVD is a special form of matrix factorization

$$D \approx UV^T$$

- Objective when D is fully observed

$$\min \left\| D - UV^T \right\|_F^2 \qquad \|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2}$$

- Objective when D is partially observed

$$\min \sum_{(i,j)\in\Omega} \left(D_{ij} - \overline{U_i}^T \overline{V_j} \right)^2$$

Ω is the set of observed cells

Non-negative, regularized matrix factorization

- Matrix factorization $D \approx UV^T$

  Objective:

$$\min \sum_{(i,j)\in\Omega} \left(D_{ij} - \overline{U_i}^T \overline{V_j}\right)^2 + \lambda \left(\|U\|_F^2 + \|V\|_F^2\right)$$

  Ω is the set of observed cells in the matrix

  $\mathrm{U} \geq 0, \mathrm{V} \geq 0$

Example 1: grocery shopping

Example: grocery shopping

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0	1	0	1	2	2
Fruits	2	3	1	1	2	2
Sweets	1	1	1	0	1	1
Bread	0	2	3	4	1	1
Coffee	0	0	0	0	1	0

- This purchase history indicates the number of time each person has purchased an item
- For clarity we're dealing with categories of items, but they can be the items themselves

This example (2018) by Piotr Gabrys

In Python

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0	1	0	1	2	2
Fruits	2	3	1	1	2	2
Sweets	1	1	1	0	1	1
Bread	0	2	3	4	1	1
Coffee	0	0	0	0	1	0

Python code

```
V = np.array(
          [[0,1,0,1,2,2],
          [2,3,1,1,2,2],
          [1,1,1,0,1,1],
          [0,2,3,4,1,1],
          [0,0,0,0,1,0]])

V = pd.DataFrame(V, columns=['John', 'Alice',
'Mary', 'Greg', 'Peter', 'Jennifer'])

V.index = ['Vegetables', 'Fruits', 'Sweets',
'Bread', 'Coffee']
```

This example (2018) by Piotr Gabrys

Matrix factorization ($V \simeq WH$)

Matrix W (items x factors) with possible names for each factor added for legibility

	Fruits pickers	Bread eaters	Veggies
Vegetables	0.00	0.04	2.74
Fruits	1.93	0.15	0.47
Sweets	0.97	0.00	0.00
Bread	0.00	2.66	1.18
Coffee	0.00	0.00	0.59

Python code

```
from sklearn.decomposition
import NMF
nmf = NMF(3)
nmf.fit(V)

H =
pd.DataFrame(np.round(nmf.compon
ents_,2), columns=V.columns)
H.index = ['Fruits pickers',
'Bread eaters',  'Veggies']

W =
pd.DataFrame(np.round(nmf.transf
orm(V),2), columns=H.index)
W.index = V.index
```

This example (2018) by Piotr Gabrys

Matrix W (items x factors)

	Fruits pickers	Bread eaters	Veggies
Vegetables	0.00	0.04	2.74
Fruits	1.93	0.15	0.47
Sweets	0.97	0.00	0.00
Bread	0.00	2.66	1.18
Coffee	0.00	0.00	0.59

Possible names for each factor added for legibility

Matrix H (factors x people)

	John	Alice	Mary	Greg	Peter	Jennifer
Fruits pickers	1.04	1.34	0.55	0.26	0.89	0.90
Bread eaters	0.00	0.60	1.12	1.36	0.03	0.07
Veggies	0.00	0.35	0.00	0.34	0.77	0.69

This example (2018) by Piotr Gabrys

Reconstruction

Original matrix (V)

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0	1	0	1	2	2
Fruits	2	3	1	1	2	2
Sweets	1	1	1	0	1	1
Bread	0	2	3	4	1	1
Coffee	0	0	0	0	1	0

Reconstructed matrix (W H)

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0.00	0.98	0.04	0.99	2.11	1.89
Fruits	2.01	2.84	1.23	0.87	2.08	2.07
Sweets	1.01	1.30	0.53	0.25	0.86	0.87
Bread	0.00	2.01	2.98	4.02	0.99	1.00
Coffee	0.00	0.21	0.00	0.20	0.45	0.41

```
reconstructed = pd.DataFrame(np.round(np.dot(W,H),2), columns=V.columns)
reconstructed.index = V.index
```

This example (2018) by Piotr Gabrys

Recommendation

Original matrix (V)

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0	1	0	1	2	2
Fruits	2	3	1	1	2	2
Sweets	1	1	1	0	1	1
Bread	0	2	3	4	1	1
Coffee	0	0	0	0	1	0

Reconstructed matrix (W H)

	John	Alice	Mary	Greg	Peter	Jennifer
Vegetables	0.00	0.98	0.04	0.99	2.11	1.89
Fruits	2.01	2.84	1.23	0.87	2.08	2.07
Sweets	1.01	1.30	0.53	0.25	0.86	0.87
Bread	0.00	2.01	2.98	4.02	0.99	1.00
Coffee	0.00	0.21	0.00	0.20	0.45	0.41

If you were to recommend one product to someone, what would you recommend and to whom?

This example (2018) by Piotr Gabrys

Evaluation

Direct evaluation

- Randomized controlled experiment
  - Renamed A/B testing for … reasons
  - People are split randomly in control/experimental
  - Control group: receives one type of recommendation
  - Experimental group: receives another type
- Metrics such as CTR, retention, etc.
- Requires infrastructure, users, policies

Evaluating with existing data

movies

users

1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

Mining of Massive Datasets 2nd edition (2014) by Leskovec et al. (Chapter 9) - slides A, B

Evaluating with existing data

movies

users

1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			?		?
				?	
	2	1			?
	3			?	
1					

Test Data Set

Mining of Massive Datasets 2nd edition (2014) by Leskovec et al. (Chapter 9) - slides A, B

Evaluation metrics

- RMSE (root of mean of squared errors)

$$\sqrt{E[(x - \hat{x})^2]}$$

- Precision @ k
  - % of recommendations that are correct among those in the top k positions
- Rank correlation
  - Spearman's correlation between system and user

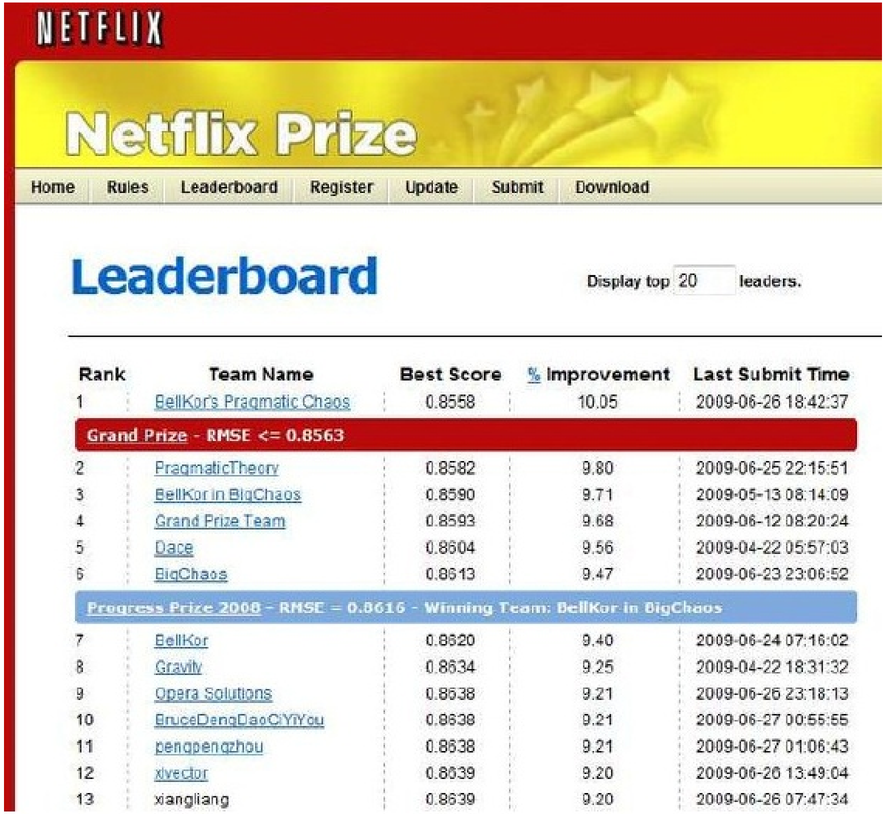
Evaluating is hard

- Accuracy is not all
- We also want diversity
- We want to be contextually sensitive
- The order of predictions matters
- RMSE might penalize a method that does well for high ratings but bad for others

Example 2: Netflix prize

Example 2: Netflix prize (2009)

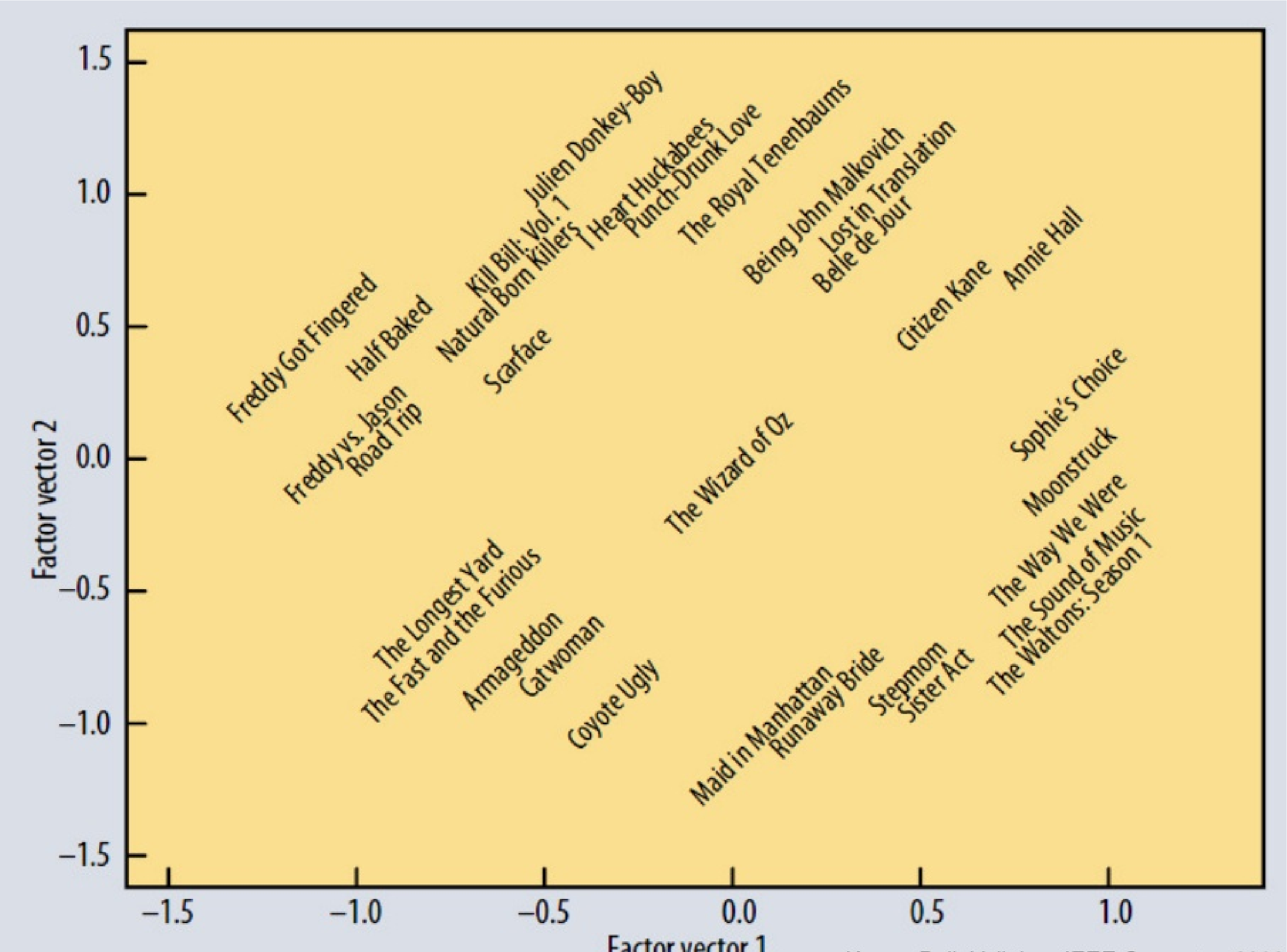
- Netflix offered $1,000,000 to anyone beating their algorithm by 10% in **RMSE**
- Provided 100M (user,movie) ratings for training
- Held a testing set and allowed one guess/day on the testing set to create a leader board



Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	BellKor's Pragmatic Chaos	0.8558	10.05	2009-06-26 18:42:37
Grand Prize - RMSE <= 0.8563				
2	PragmaticTheory	0.8582	9.80	2009-06-25 22:15:51
3	BellKor in BigChaos	0.8590	9.71	2009-05-13 08:14:09
4	Grand Prize Team	0.8593	9.68	2009-06-12 08:20:24
5	Dace	0.8604	9.56	2009-04-22 05:57:03
6	BigChaos	0.8613	9.47	2009-06-23 23:06:52
Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos				
7	BellKor	0.8620	9.40	2009-06-24 07:16:02
8	Gravity	0.8634	9.25	2009-04-22 18:31:32
9	Opera Solutions	0.8638	9.21	2009-06-26 23:18:13
10	BruceDengDaoCiYiYou	0.8638	9.21	2009-06-27 00:55:55
11	pengpengzhou	0.8638	9.21	2009-06-27 01:06:43
12	xivector	0.8639	9.20	2009-06-26 13:49:04
13	xiangliang	0.8639	9.20	2009-06-26 07:47:34

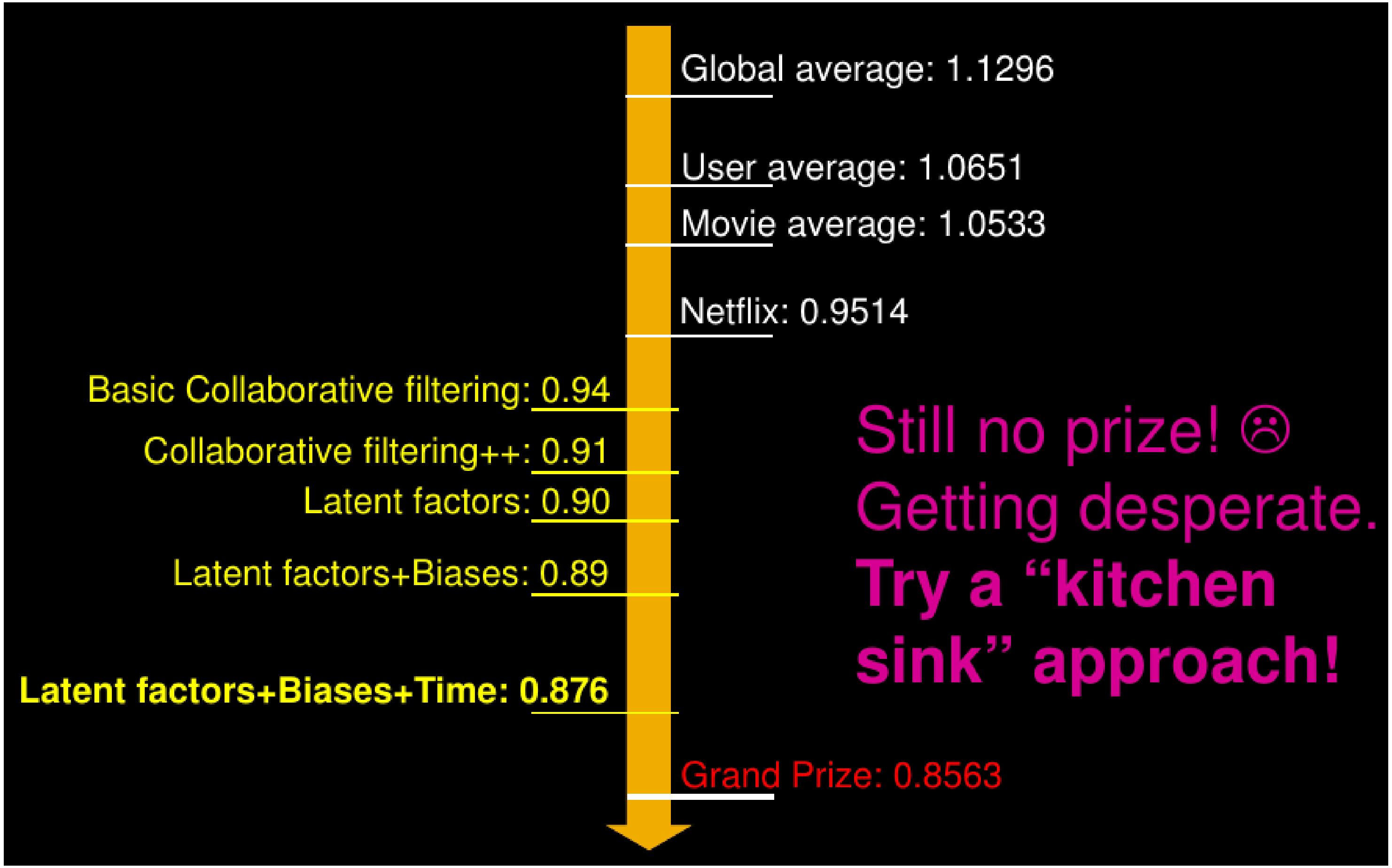
Latent factors

In latent factor space, similar movies are mapped to similar points



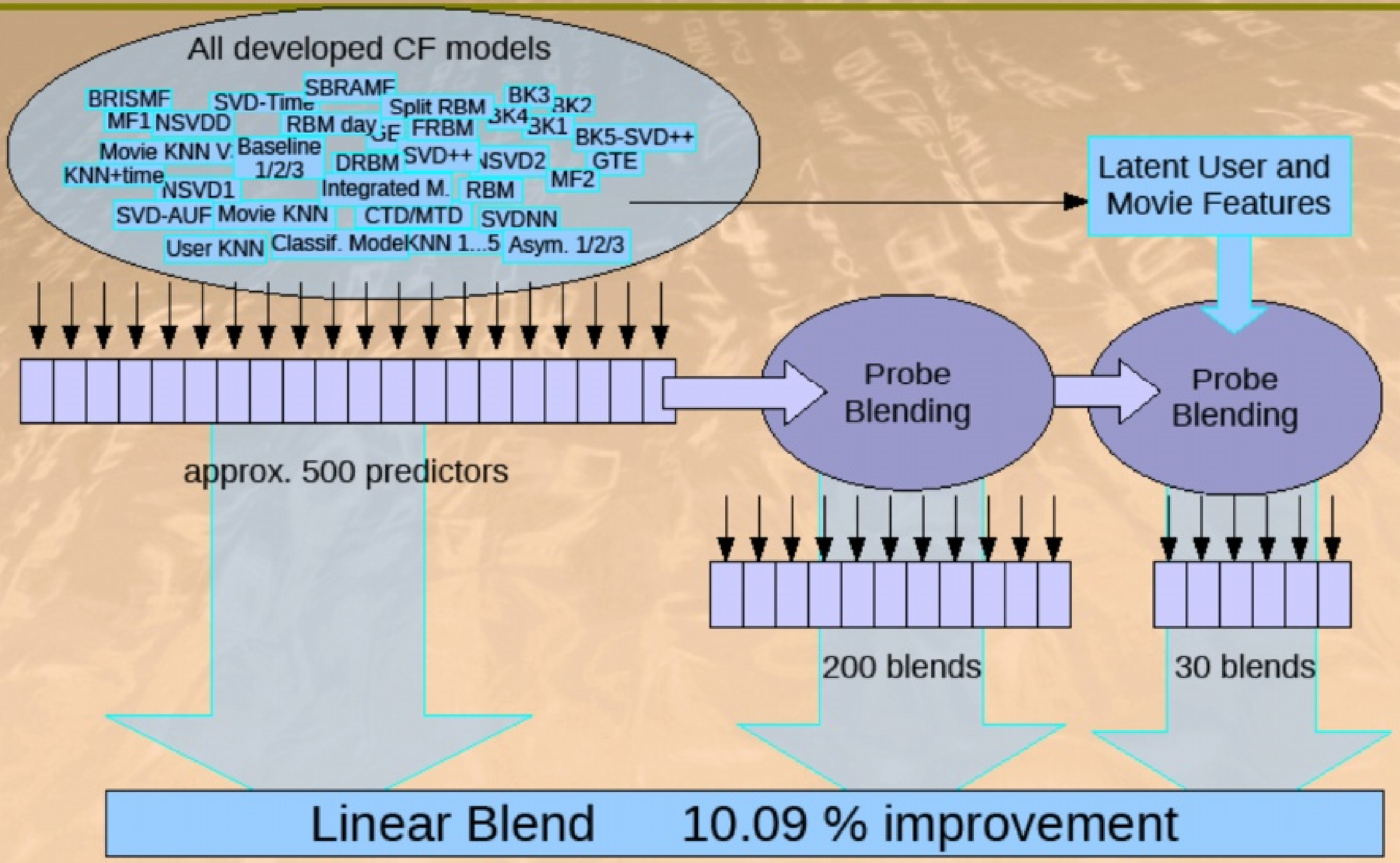
Koren, Bell, Volinksy, IEEE Computer, 2009

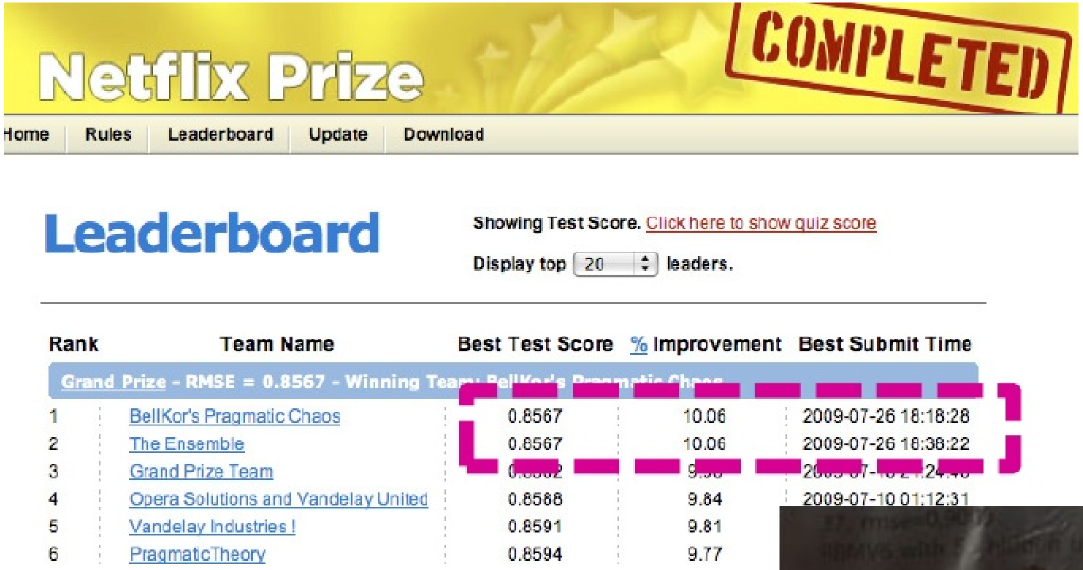
Shortly before deadline …



The big picture

Solution of BellKor's Pragmatic Chaos





Netflix Prize — COMPLETED

Home | Rules | Leaderboard | Update | Download

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	[illegible]	[illegible]	[illegible]
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	
6	PragmaticTheory	0.8594	9.77	

26 July 2009.- Bellkor team submits 40 minutes before the deadline, “The Ensemble” team made of a mix of other teams submitted 20 minutes before the deadline.

Bellkor team wins one million dollars

Summary

Things to remember

- Content-based recommendations
- Interaction-based recommendations
  - User-based
  - Item-based
  - Latent factors based
- Evaluation methods

Exercises on this topic

- Mining of Massive Datasets 2nd edition (2014) by Leskovec et al. Note that some exercises cover advanced concepts:
  - Exercises 9.2.8
  - Exercises 9.3.4
  - Exercises 9.4.6