CHULA ENGINEERING
Foundation toward Innovation

COMPUTER

Chula Big Data and IoT
Center of Excellence
(CUBIC)

DataMind
Data Mining Group
Machine Intelligence and Knowledge Discovery Lab
Chulalongkorn University

Data Analytics (Part2)
Python for Data Analytics

Peerapon Vateekul, Ph.D.
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University
Peerapon.v@chula.ac.th

Outlines

- Lab (cont.)
- Collaborative Filtering

PIERIAN DATA

https://www.pieriandata.com/

- Understand data analytics **tasks**
- Be able to identify tasks and **tools** (technique) from a given problem

Lab (cont.)

Lab9: Association Rule
Online Retail Data Set

- A useful (but somewhat overlooked) technique is called association analysis which attempts to find common patterns of items in large data sets. One specific application is often called market basket analysis.

- 541909 rows and 8 columns

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

- Library

```
# Build up the frequent items
frequent_itemsets = apriori(basket_sets, min_support=0.07, use_colnames=True)
```

```
# Create the rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules
```

- Output image

```
rules[ (rules['lift'] >= 6) &
       (rules['confidence'] >= 0.8) ]
```

	antecedants	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
2	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.096939	0.094388	0.079082	0.815789	8.642959	0.069932	4.916181
3	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.094388	0.096939	0.079082	0.837838	8.642959	0.069932	5.568878
17	(SET/6 RED SPOTTY PAPER PLATES)	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.127551	0.132653	0.102041	0.800000	6.030769	0.085121	4.336735
18	(SET/6 RED SPOTTY PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES)	0.137755	0.127551	0.122449	0.888889	6.968889	0.104878	7.852041

Reference: http://pbpython.com/market-basket-analysis.html

Lab10: Time Series Forecasting with ARIMA: *CO2 Data Set*

- We'll be working with a dataset called "**Atmospheric CO2** from Continuous Air Samples at Mauna Loa Observatory, Hawaii, U.S.A.," which collected CO2 samples from March 1958 to December 2001.1969 rows and 2 columns

	co2
1958-03-29	316.1
1958-04-05	317.3
1958-04-12	317.6
1958-04-19	317.5
1958-04-26	316.4

```
1958-03-01    316.100000
1958-04-01    317.200000
1958-05-01    317.433333
1958-06-01    315.625000
1958-07-01    315.625000
Freq: MS, Name: co2, dtype: float64
```

- Library

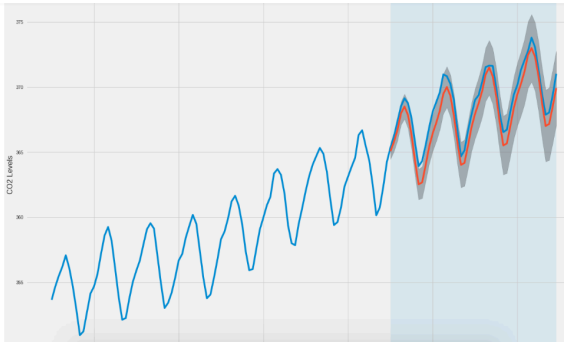
Fitting an ARIMA Time Series Model

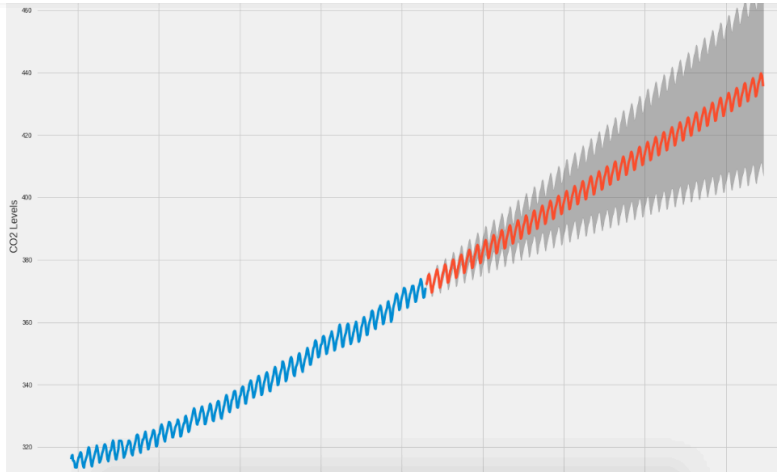
```
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 1, 12),
                                enforce_stationarity=False, # Whether or not
                                enforce_invertibility=False) # Whether or not

results = mod.fit()

print(results.summary())
```

- Output image

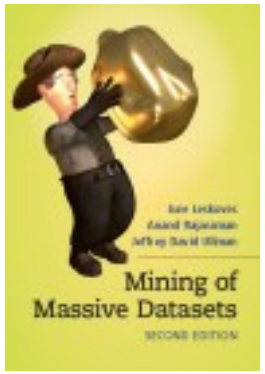




Recommender System: Collaborative Filtering

Acknowledgement

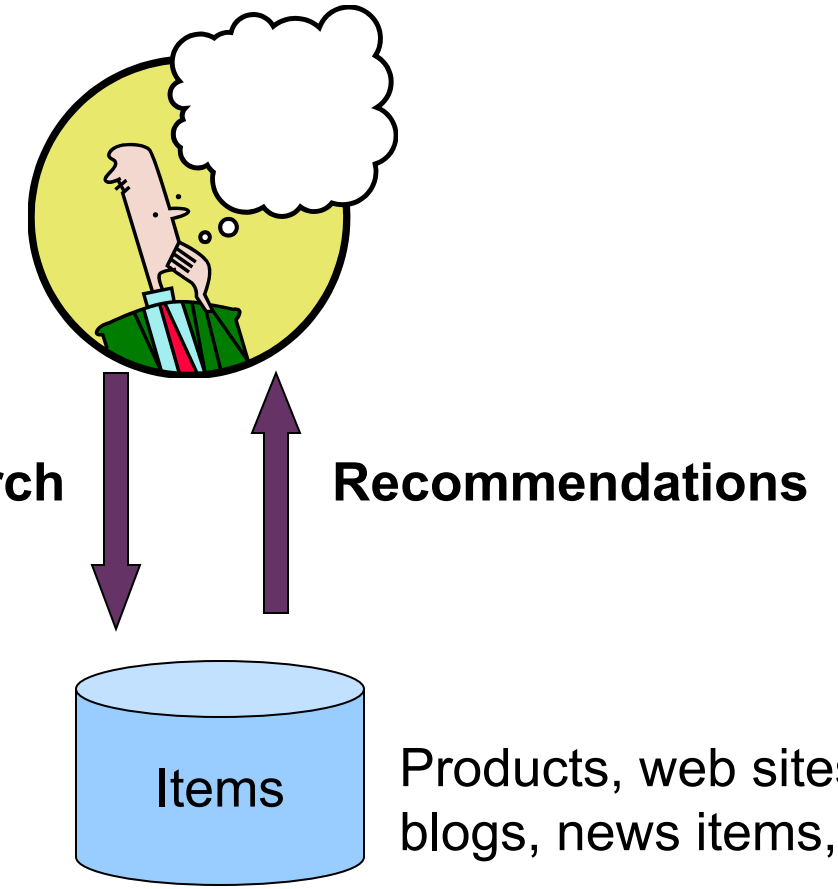
- **Jure Leskovec, Anand Rajaraman, Jeff Ullman**
- Stanford University
- Mining of Massive Datasets
- www.mmds.org



- **Prof. Dr. Boonserm Kijsirikul**
- 2110746 Big Data Analytics

Introduction to Recommender System

Recommendations



Products, web sites, blogs, news items, …

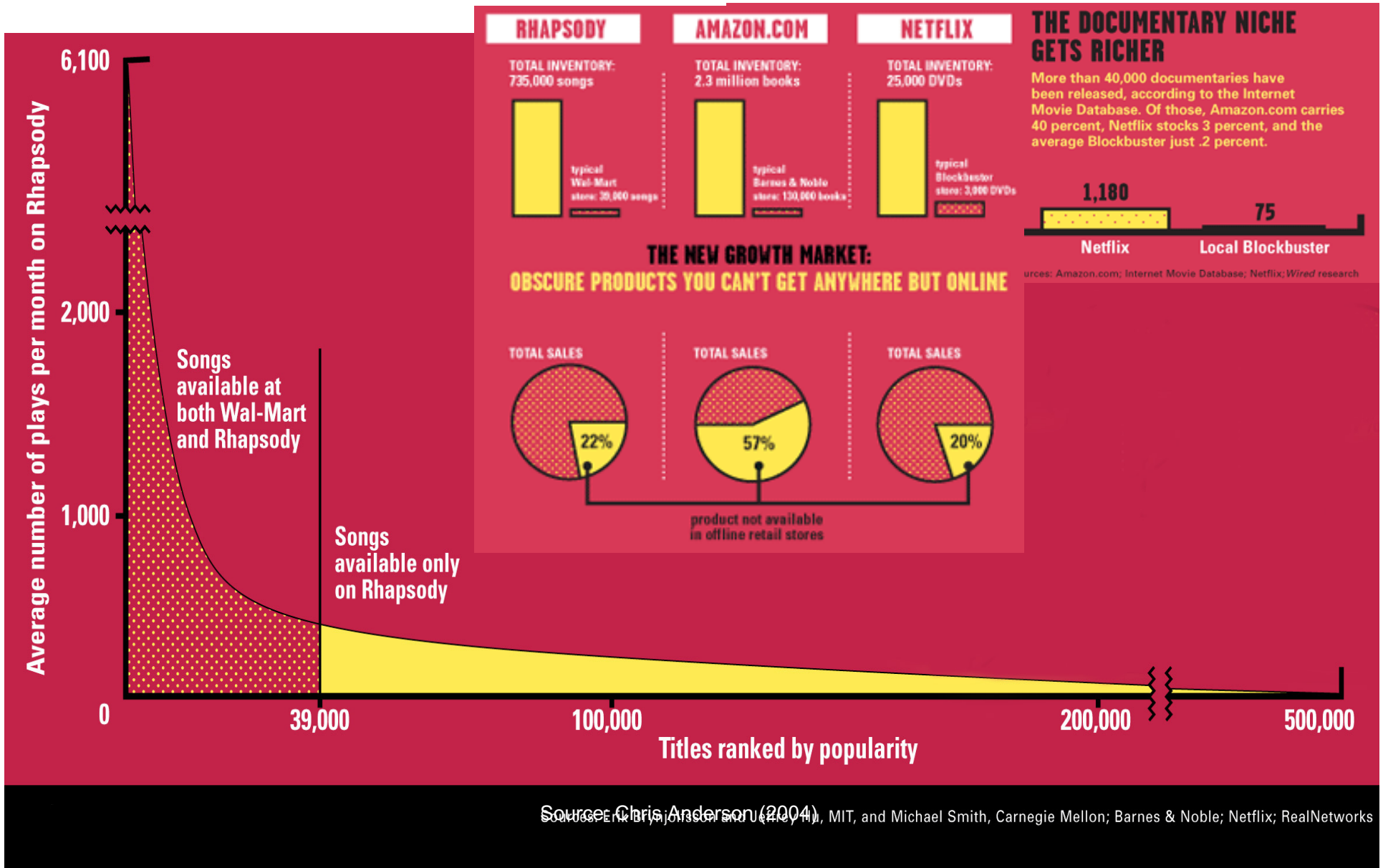
Examples:

amazon.com
PANDORA
StumbleUpon
del.icio.us
NETFLIX
movielens
helping you find the *right* movies
last·fm
the social music revolution
Google News
YouTube
XBOX LIVE

From Scarcity to Abundance

- **Shelf space is a scarce commodity for traditional retailers**
  - Also: TV networks, movie theaters,…
- **Web enables near-zero-cost dissemination of information about products**
  - From scarcity to abundance
- **More choice necessitates better filters**
  - Recommendation engines
  - How **Into Thin Air** made **Touching the Void** a bestseller: http://www.wired.com/wired/archive/12.10/tail.html

Sidenote: The Long Tail



Source: Chris Anderson (2004)

Types of Recommendations

- **Editorial and hand curated**
  - List of favorites
  - Lists of "essential" items
- **Simple aggregates**
  - Top 10, Most Popular, Recent Uploads
- **Tailored to individual users**
  - Amazon, Netflix, …

How to recommend to each user?

- 1) Content-based
- **2) Collaborative**
- 3) Latent factor based

Today!

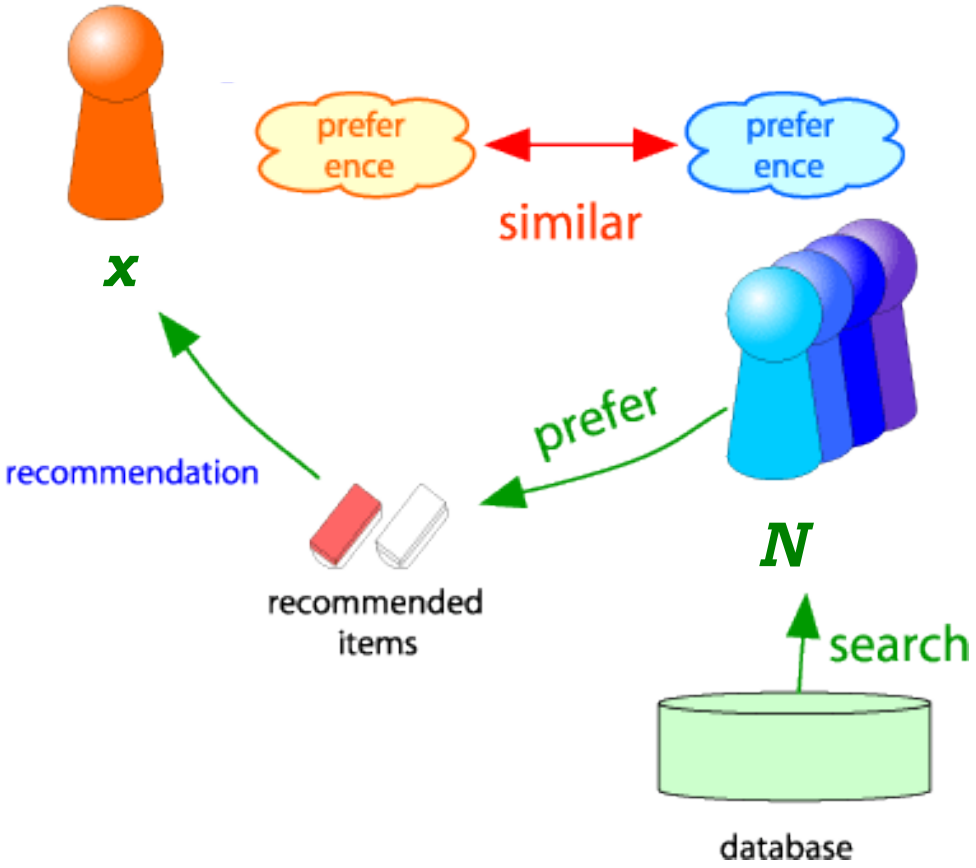
Collaborative Filtering (CF)

1. User-based CF
2. Item-based CF

1	3	4			
	3	5			5
		4	5		5
		3			
1					

User-Based CF

- Consider user ***x***
- Find set ***N*** of other users whose ratings are "**similar**" to ***x***'s ratings
- Estimate ***x***'s ratings based on ratings of users in ***N***



Finding "Similar" Users

- Let r_x be the vector of user x's ratings
- **Jaccard similarity measure**
  - **Problem:** Ignores the value of the rating
- **Cosine similarity measure**
  - $\text{sim}(x, y) = \cos(r_x, r_y) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$
  - **Problem:** Treats missing ratings as "negative"

r_x = [*, _, _, *, ***]
r_y = [*, _, **, **, _]

r_x, r_y as sets:
r_x = {1, 4, 5}
r_y = {1, 3, 4}

r_x, r_y as points:
r_x = {1, 0, 0, 1, 3}
r_y = {1, 0, 2, 2, 0}

Similarity Metric

Cosine sim:

$$sim(x, y) = \frac{\sum_i r_{xi} \cdot r_{yi}}{\sqrt{\sum_i r_{xi}^2} \cdot \sqrt{\sum_i r_{yi}^2}}$$

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- **Intuitively we want: sim(*A*, *B*) > sim(*A*, *C*)**
- **Jaccard similarity:** 1/5 < 2/4
- **Cosine similarity:** 0.386 > **0.322**
  - Considers missing ratings as "negative"
  - **Solution: subtract the (row) mean**

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	2/3			5/3	−7/3		
B	1/3	1/3	−2/3				
C				−5/3	1/3	4/3	
D		0					0

sim A,B vs. A,C:
0.092 > -0.559

Notice cosine sim. is correlation when data is centered at 0

Rating Predictions

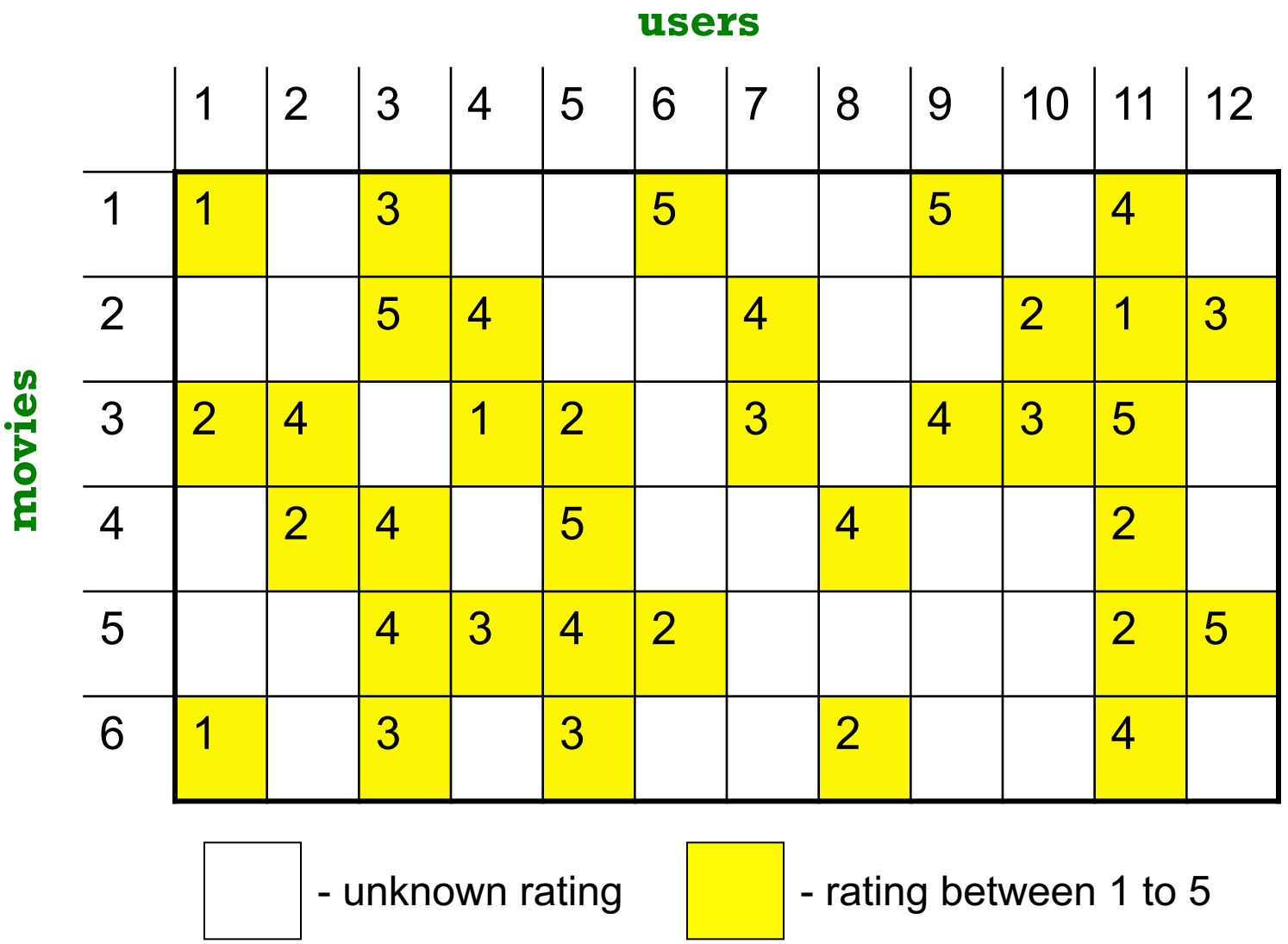
From similarity metric to recommendations:

- Let r_x be the vector of user x's ratings
- Let N be the set of k users most similar to x who have rated item i
- **Prediction for item *s* of *user x*:**
  - $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
  - $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$
  - Other options?
- **Many other tricks possible…**

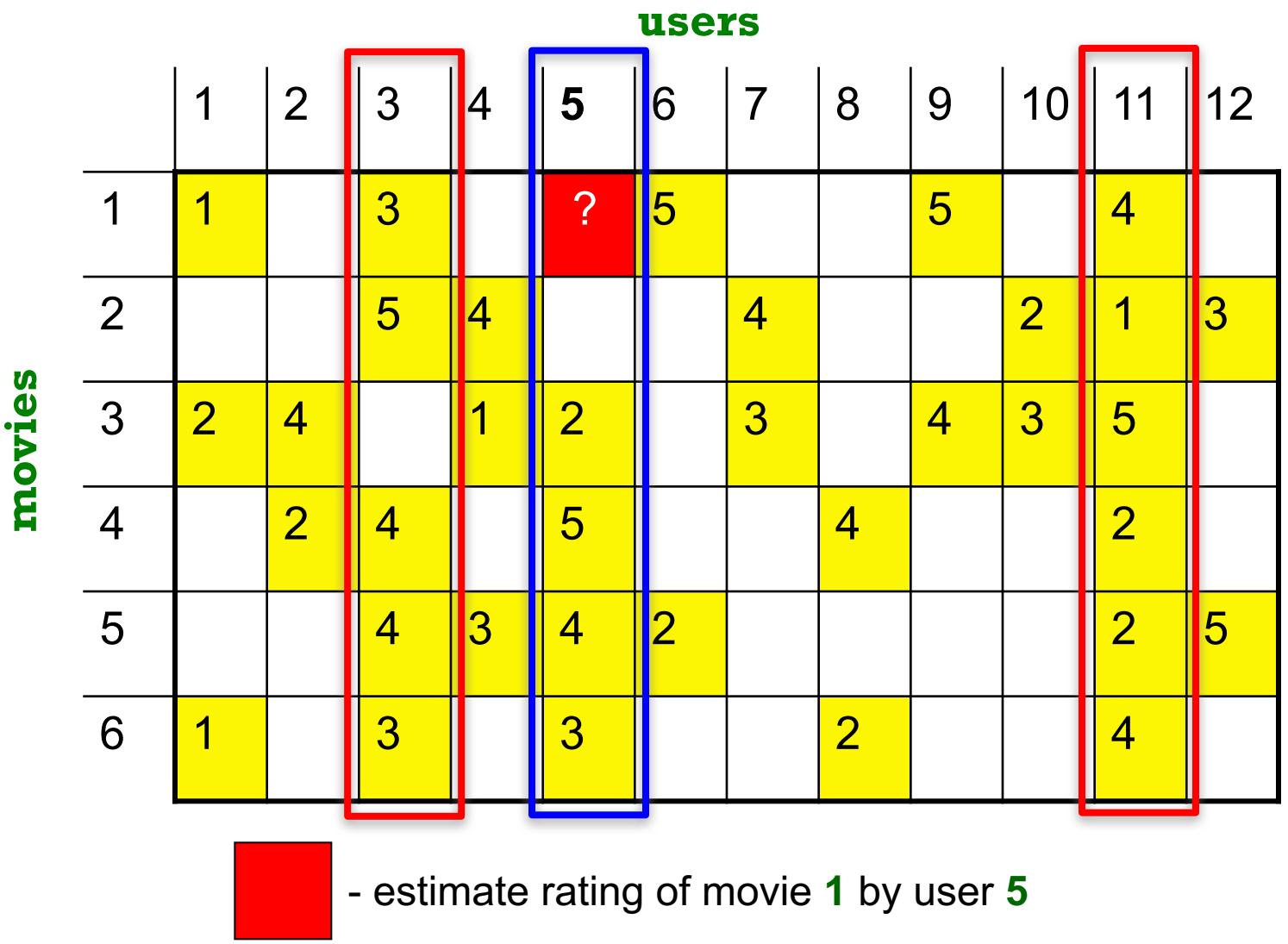
Shorthand:

$s_{xy} = sim(x, y)$

CF (|N|=2): Train


users
movies
1 2 3 4 5 6 7 8 9 10 11 12
1 1 3 5 5 4
2 5 4 4 2 1 3
3 2 4 1 2 3 4 3 5
4 2 4 5 4 2
5 4 3 4 2 2 5
6 1 3 3 2 4
- unknown rating
- rating between 1 to 5

1 User-Based CF (|N|=2): Problem



User-Based CF (|N|=2): Steps

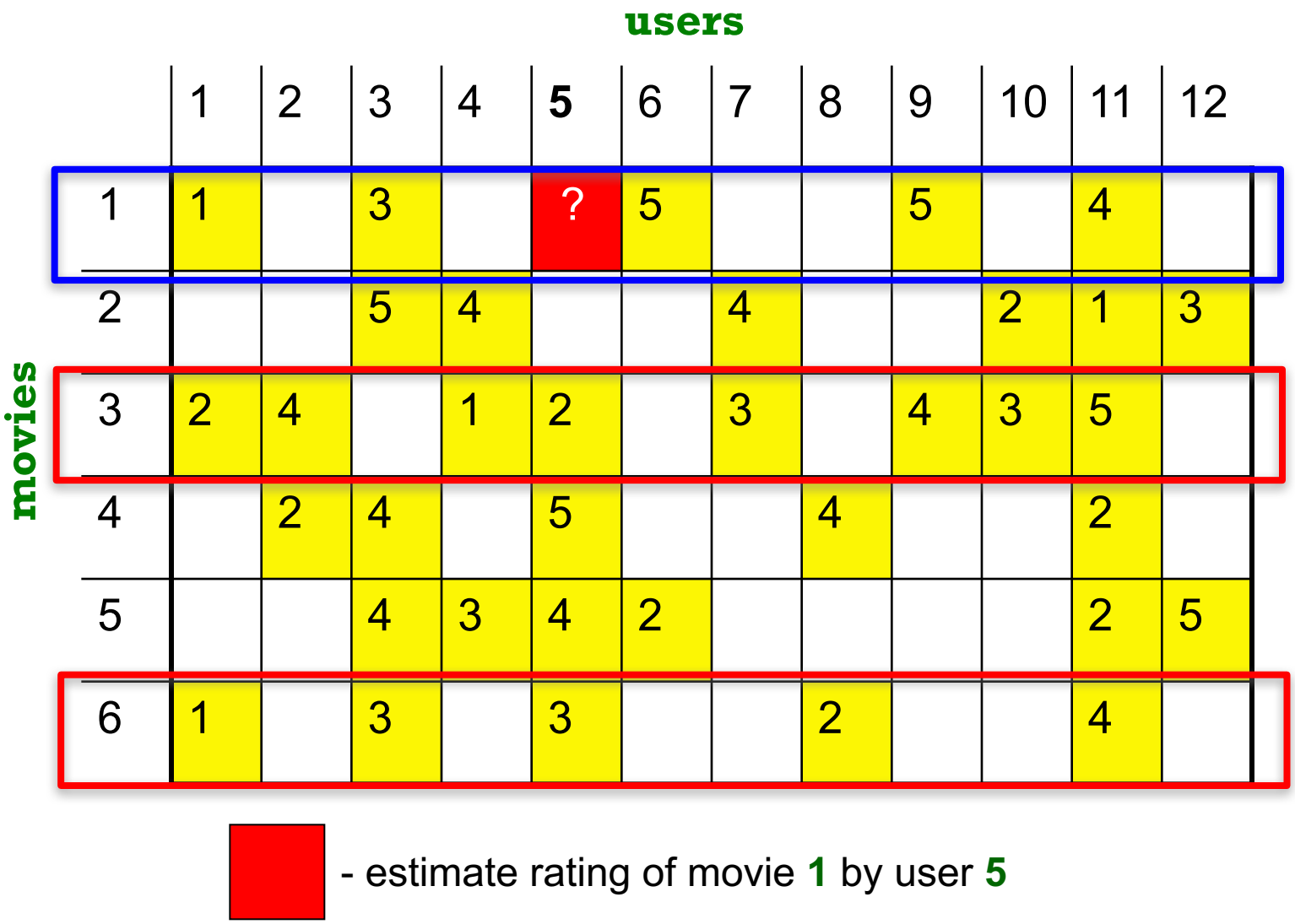
- Who are the similar users (N=2) to User5?
  - Normalize matrix by "an average of **user** rating"
  - Compute cosine similarity
  - Which **users**?
- Predict rating of Item1
  - Average=?
  - Weighted Average=?

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

2 Item-Based CF (|N|=2): Problem



Item-Based CF (|N|=2): Steps

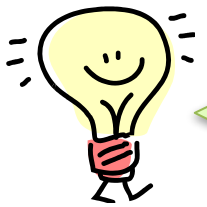
- What are the similar items (N=2) to Item1?
  - Normalize matrix by "an average of **item** rating"
  - Compute cosine similarity
  - Which **items**?

- Predict rating of User5
  - Average=?
  - Weighted Average=?

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	



- There is still a better prediction solution

Intuitive Idea

- Mean movie rating: **3.7 stars**
- *The Sixth Sense* is **0.5** stars **above** avg.
- Joe rates **0.2** stars **below** avg.
  ⇒ Baseline estimation:
  ***Joe* will rate *The Sixth Sense* 4 stars**

users

movies

	1	2	3	4	**5**	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- What should be **an expected rate** of “User5 on Item3”?
- What should be **deviation from the expectation?**

CF: Common Practice

Before:

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

- Define **similarity** s_{ij} of items ***i*** and ***j***
- Select ***k*** nearest neighbors $N(i; x)$
  - Items most similar to ***i***, that were rated by ***x***
- Estimate rating r_{xi} as the weighted average:

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

baseline estimate for r_{xi}

$$b_{xi} = \mu + b_x + b_i$$

- μ = overall mean movie rating
- b_x = rating deviation of user ***x***
  = (*avg. rating of user* ***x***) $- \mu$
- b_i = rating deviation of movie ***i***

Item-Based CF (|N|=2) Using Common Approach

- What are the similar items (N=2) to Item1?
  - Normalize matrix by "an average of **item** rating"
  - Compute cosine similarity
  - Which items?

- Predict rating of User5

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

baseline estimate for r_{xi}

$$b_{xi} = \mu + b_x + b_i$$

users

movies

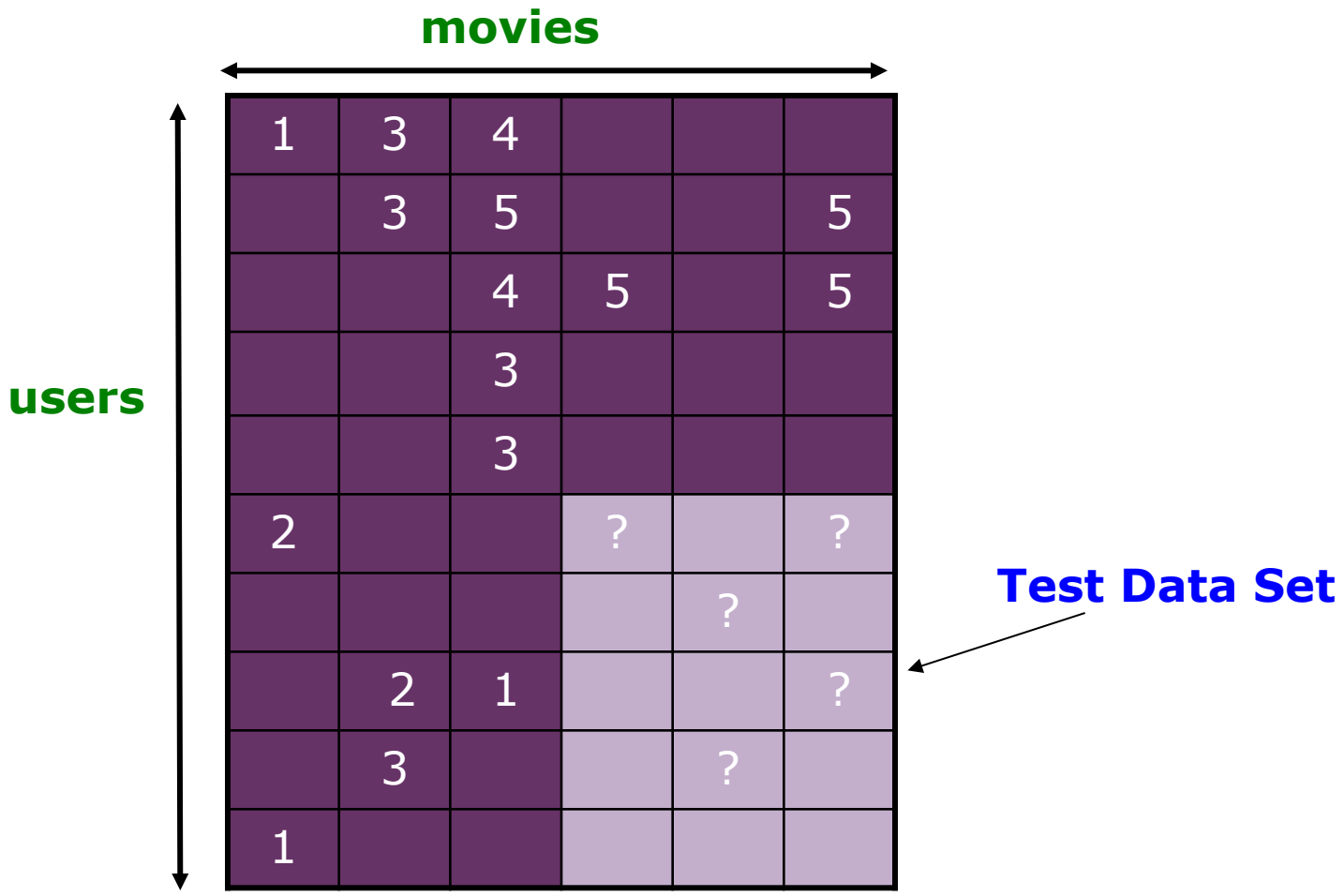
	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Remarks

- User-based CF vs. Item-based CF
  - In practice, it has been observed that item-item often works better than user-user
  - Why? Items are simpler, users have multiple tastes

- Major drawbacks in CF
  - Cold Start:
    - Need enough users in the system to find a match
  - Sparsity:
    - Hard to find users that have rated the same items
  - First rater:
    - New users or items; NO rating data to find neighbors -- cannot recommend!!!
  - Popularity bias:
    - Cannot recommend items to someone with unique taste
    - Tends to recommend popular items

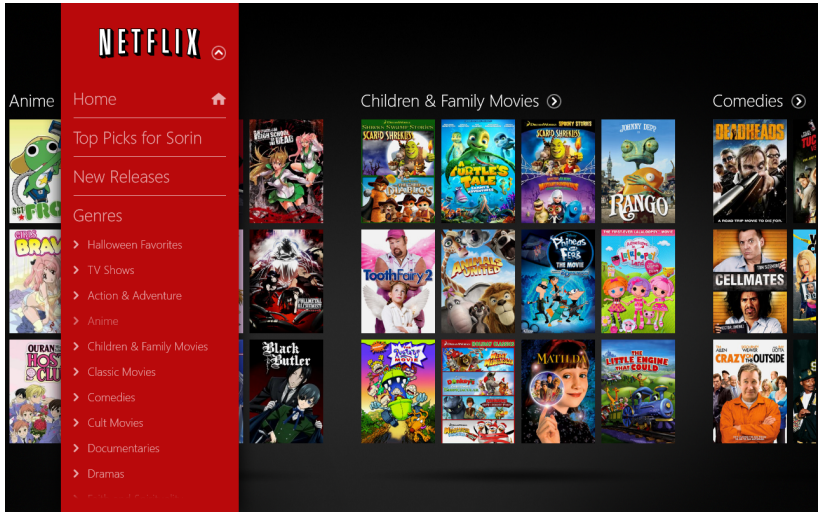
Evaluation



Evaluating Predictions

- **Compare predictions with known ratings**
  - **Root-mean-square error** (RMSE)
    - $\sqrt{\sum_{xi}(r_{xi} - r^*_{xi})^2}$ where r_{xi} is predicted, r^*_{xi} is the true rating of x on i
  - **Precision at top 10**:
    - % of those in top 10
  - **Rank Correlation**:
    - Spearman's *correlation* between system's and user's complete rankings
- **Another approach: 0/1 model**
  - **Coverage (Recall):**
    - Number of items/users for which system can make predictions
  - **Precision:**
    - Accuracy of predictions
  - **Receiver operating characteristic** (ROC)
    - Tradeoff curve between false positives and false negatives

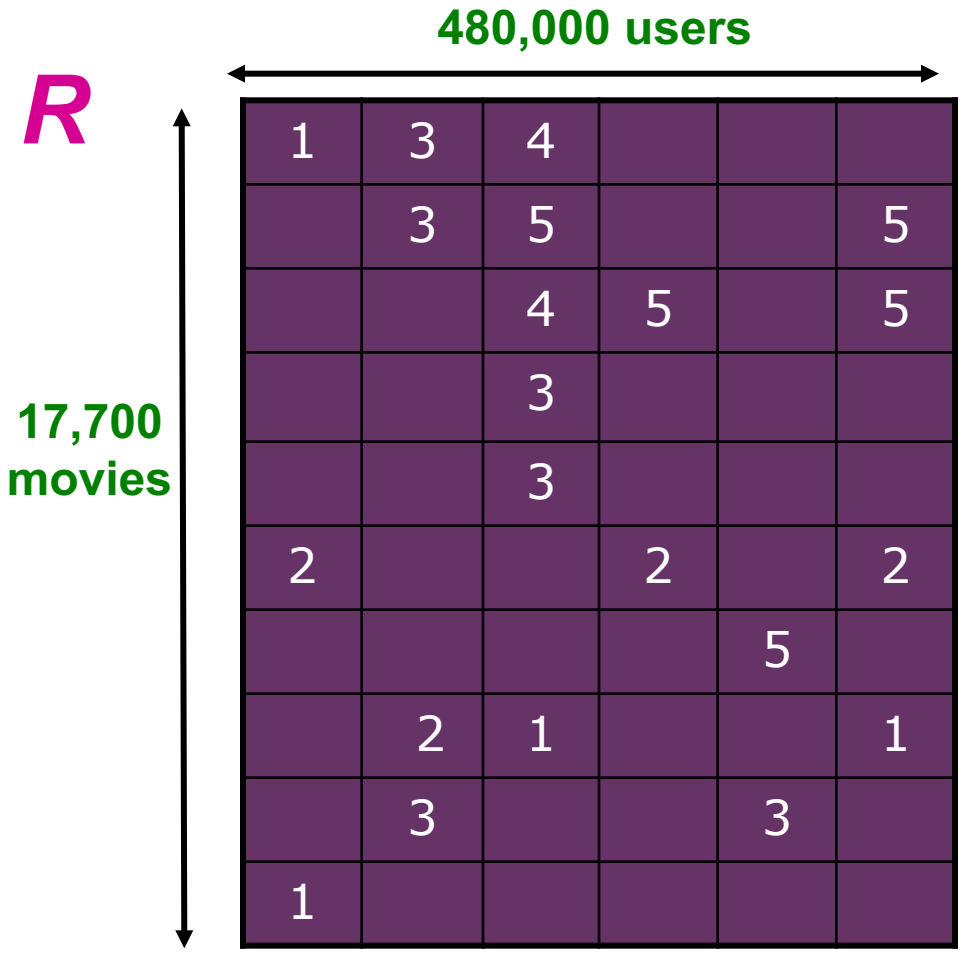
- **The Netflix Prize**
- **Further Improvement**

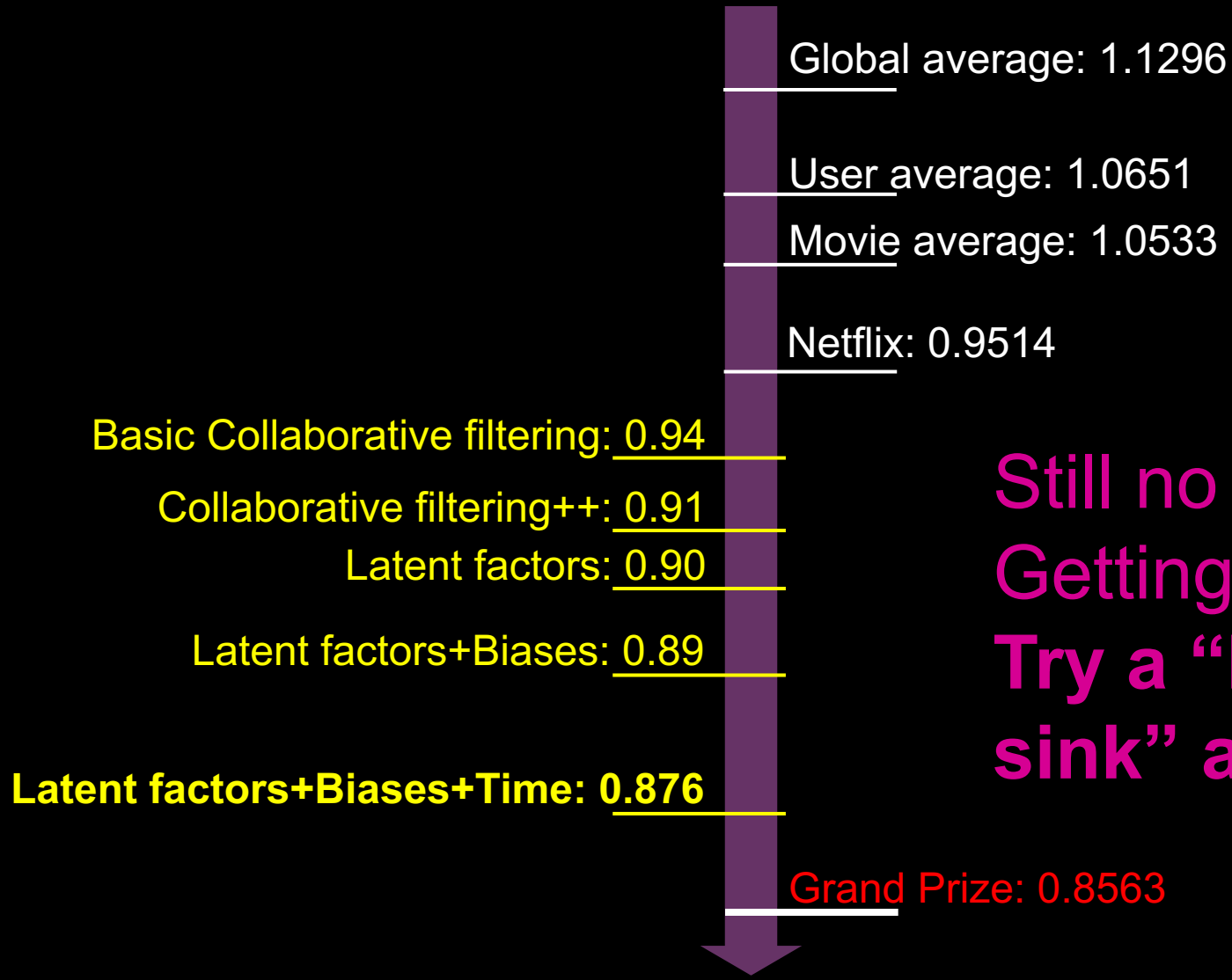


The Netflix Prize

- **Training data**
  - 100 million ratings, 480,000 users, 17,770 movies
  - 6 years of data: 2000-2005
- **Test data**
  - Last few ratings of each user (2.8 million)
  - **Evaluation criterion:** Root Mean Square Error (RMSE) $= \frac{1}{|R|}\sqrt{\sum_{(i,x)\in R}(\hat{r}_{xi} - r_{xi})^2}$
  - **Netflix's system RMSE: 0.9514**
- **Competition**
  - 2,700+ teams
  - **$1 million** prize for 10% improvement on Netflix

The Netflix Utility Matrix *R*



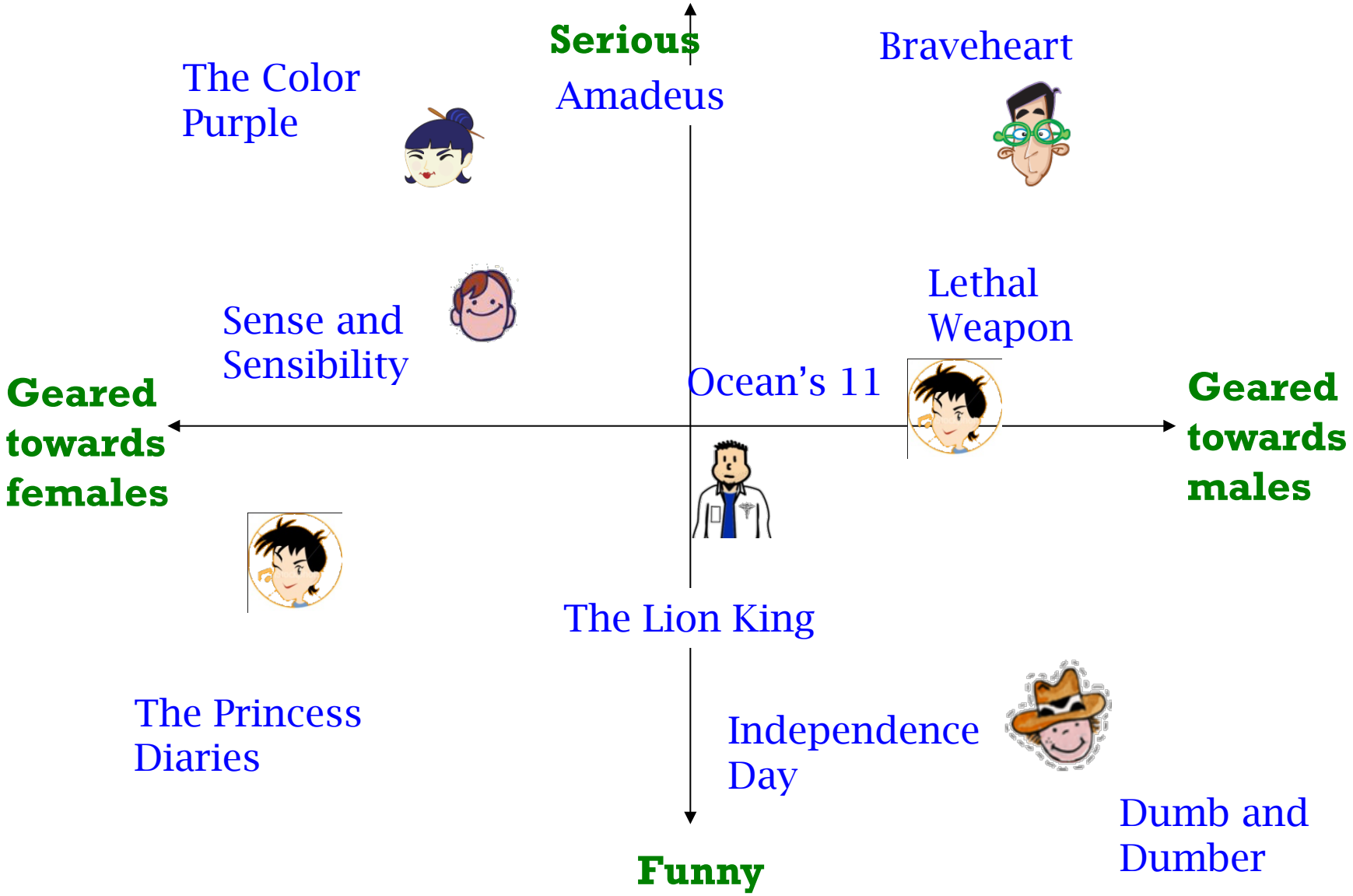


Still no prize! ☹
Getting desperate.
Try a "kitchen sink" approach!

Further Improvement

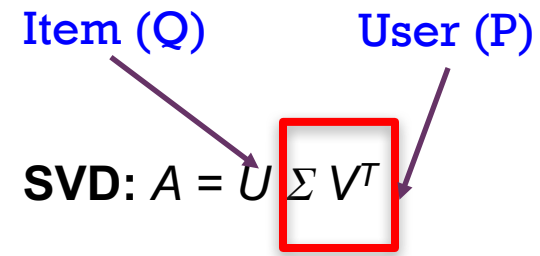
- Latent Factor
- Temporal Model
- Ensemble

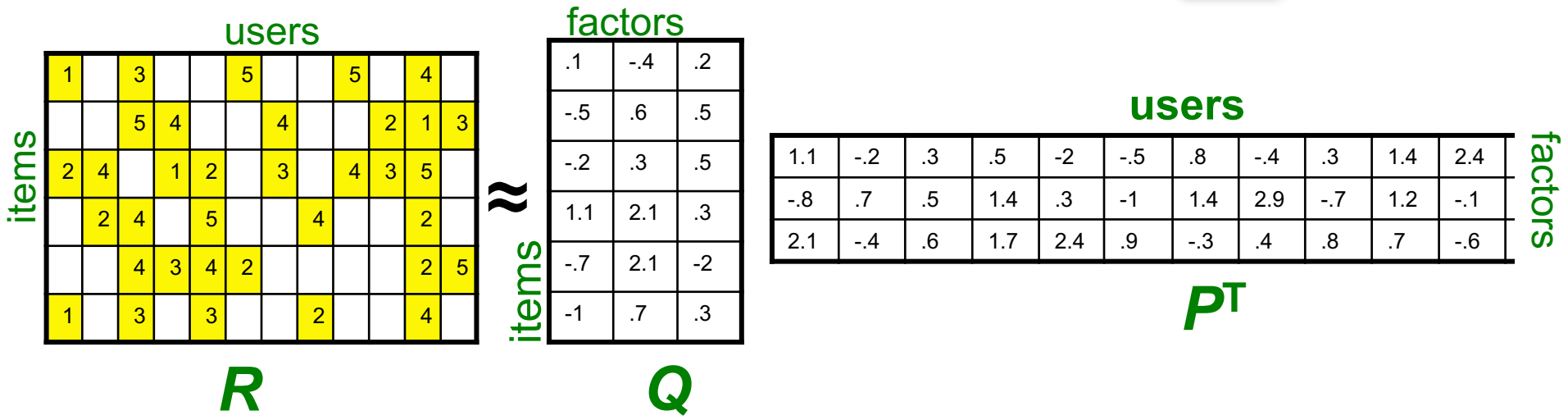
Latent Factor Models (e.g., SVD)



Latent Factor Models

- "SVD" on Netflix data: $\mathbf{R} \approx \boldsymbol{Q} \cdot \boldsymbol{P}^T$





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

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6

Matrix Factorization Techniques for Recommender Systems

Authors: Yehuda Koren Yahoo Research
Robert Bell AT&TLabs
Chris Volinsky AT&TLabs

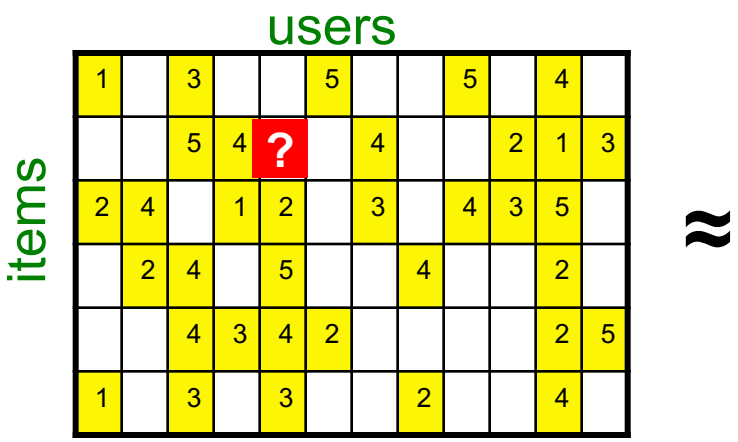
Published in:
· Journal
Computer archive
Volume 42 Issue 8, August 2009
Pages 30-37
IEEE Computer Society Press Los Alamitos, CA, USA

2009 Article
- orig-research

Bibliometrics
· Downloads (6 Weeks): 0
· Downloads (12 Months): 0
· Downloads (cumulative): 0
· Citation Count: 569

Ratings as Products of Factors

- **How to estimate the missing rating of user *x* for item *i*?**



≈

$$\hat{r}_{xi} = q_i \cdot p_x = \sum_f q_{if} \cdot p_{xf}$$

q_i = row i of Q
p_x = column x of P^T

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

items × factors = Q

•

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

factors × users = P^T

Ratings as Products of Factors

- **How to estimate the missing rating of user *x* for item *i*?**

users

items

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

≈

$$\hat{r}_{xi} = q_i \cdot p_x$$

$$= \sum_f q_{if} \cdot p_{xf}$$

q_i = row i of Q

p_x = column x of P^T

.1	-.4	.2
-.5	**.6**	**.5**
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

items

factors

Q

•

users

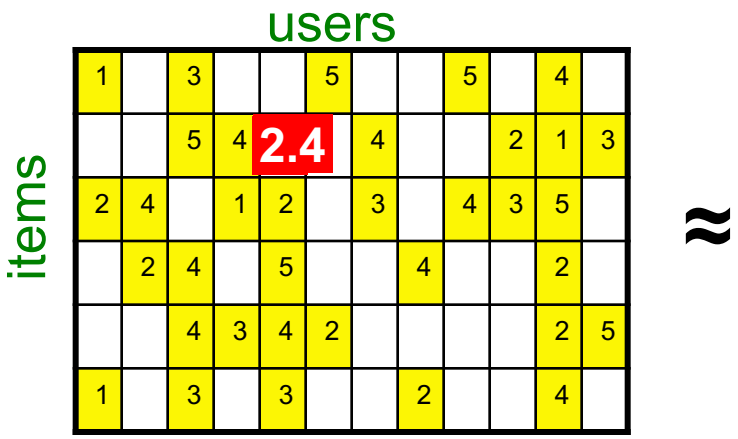
factors

1.1	-.2	.3	.5	**-2**	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	**.3**	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	**2.4**	.9	-.3	.4	.8	.7	-.6	.1

P^T

Ratings as Products of Factors

- **How to estimate the missing rating of user *x* for item *i*?**



≈

$$\hat{r}_{xi} = q_i \cdot p_x = \sum_f q_{if} \cdot p_{xf}$$

q_i = row i of Q
p_x = column x of P^T

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

items

f factors

Q

•

users

f factors

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

P^T

Temporal Biases & Factors

- **Original model:**
  $r_{xi} = \mu + b_x + b_i + q_i \cdot p_x$

- **Add time dependence to biases:**
  $r_{xi} = \mu + b_x(t) + b_i(t) + q_i \cdot p_x$
  - Make parameters b_x and b_i to depend on time
  - **(1)** Parameterize time-dependence by linear trends
    (2) Each bin corresponds to 10 consecutive weeks

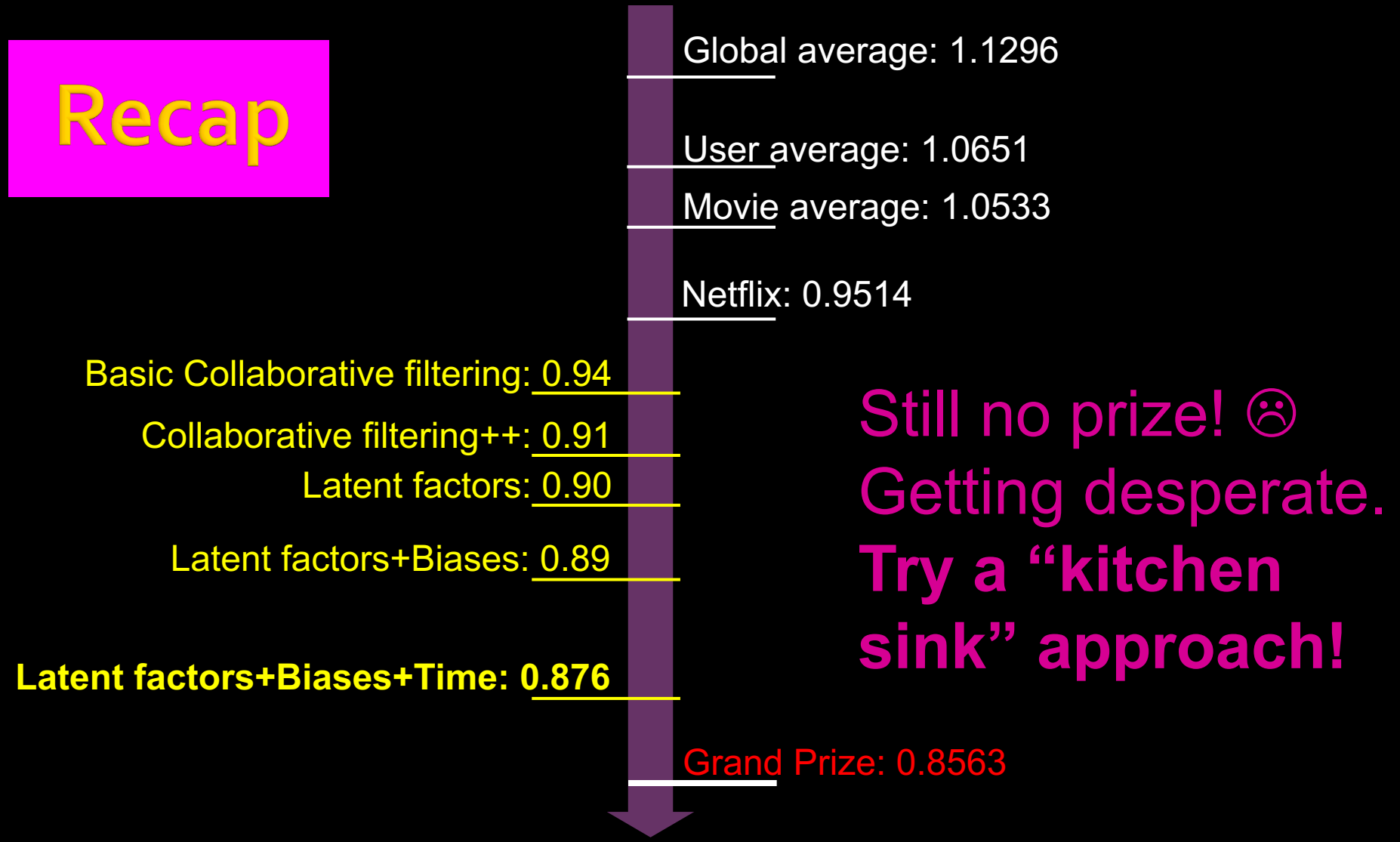
$$b_i(t) = b_i + b_{i,\mathrm{Bin}(t)}$$

- **Add temporal dependence to factors**
  - $p_x(t)$… user preference vector on day t

P = user matrix
Q = item matrix
B_x = user bias
B_i = item bias

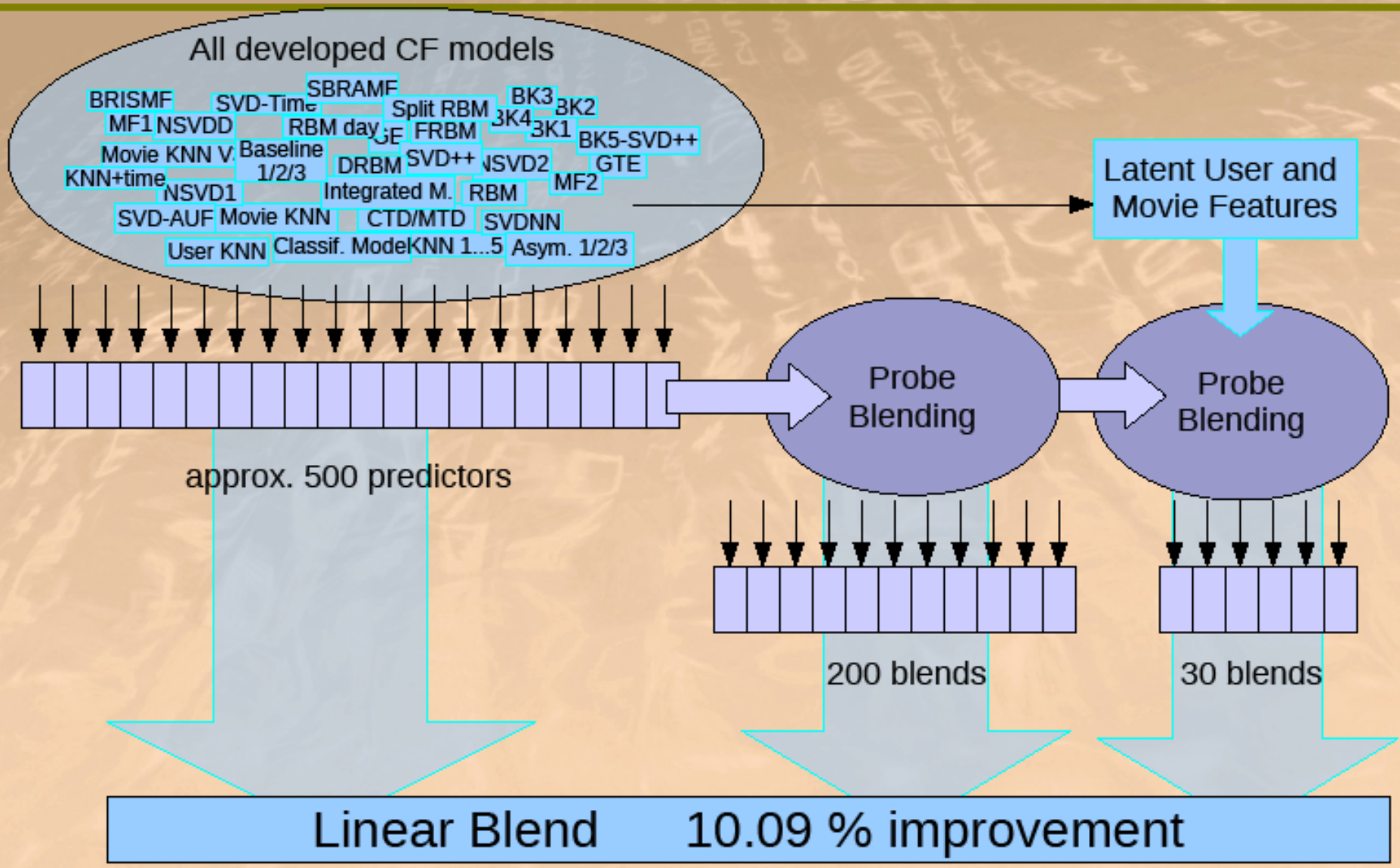
Recap: SVD

Recap



Still no prize! ☹
Getting desperate.
Try a "kitchen sink" approach!

The big picture
Solution of BellKor's Pragmatic Chaos



NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update | Download

Leaderboard

Showing Test Score. Click here to show quiz score

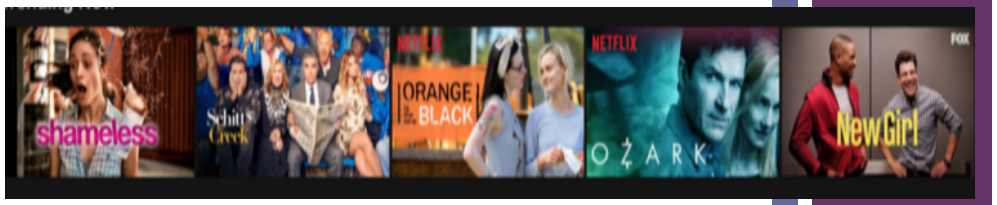
Display top 20 leaders.

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	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8642	9.27	2009-07-15 14:53:22
14	Gravity	0.8643	9.26	2009-04-22 18:31:32
15	Ces	0.8651	9.18	2009-06-21 19:24:53
16	Invisible Ideas	0.8653	9.15	2009-07-15 15:53:04
17	Just a guy in a garage	0.8662	9.06	2009-05-24 10:02:54
18	J Dennis Su	0.8666	9.02	2009-03-07 17:16:17
19	Craig Carmichael	0.8666	9.02	2009-07-25 16:00:54
20	acmehill	0.8668	9.00	2009-03-21 16:20:50
Progress Prize 2007 - RMSE = 0.8723 - Winning Team: KorBell				

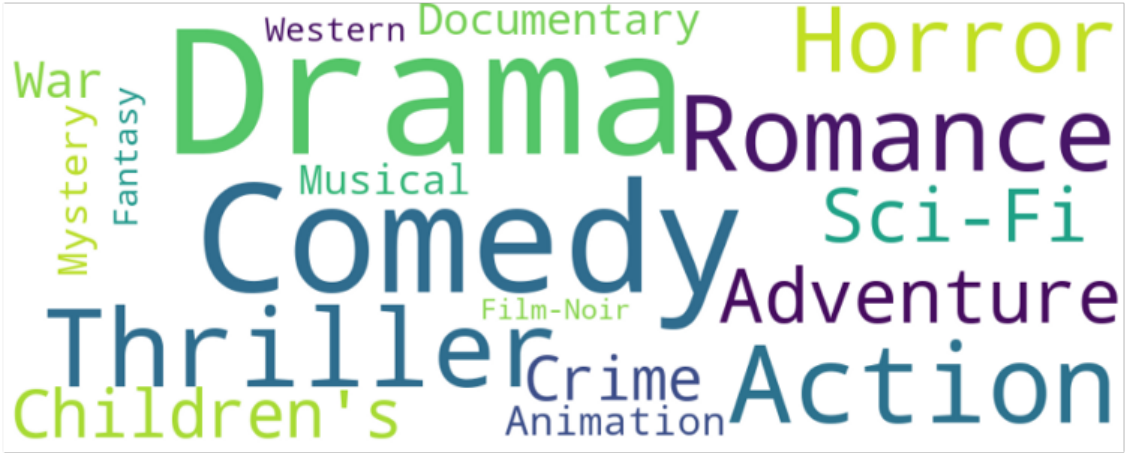
Million $ Awarded Sept 21st 2009

BellKor's Team

Lab11: Recommender System
Movie Lens Data Set



- Predict movie rating
  - User-based CF
  - Item-based CF
  - SVD



- Library

```
import scipy.sparse as sp
from scipy.sparse.linalg import svds

#get SVD components from train matrix. Choose k.
u, s, vt = svds(train_data_matrix, k = 20)
s_diag_matrix=np.diag(s)
X_pred = np.dot(np.dot(u, s_diag_matrix), vt)
print('User-based CF MSE: ' + str(rmse(X_pred, test_data_matrix)))
```

```
User-based CF MSE: 2.715056956440157
```

- Output image

```
In [0]: from sklearn.metrics import mean_squared_error
        from math import sqrt
        def rmse(prediction, ground_truth):
            prediction = prediction[ground_truth.nonzero()].flatten()
            ground_truth = ground_truth[ground_truth.nonzero()].flatten()
            return sqrt(mean_squared_error(prediction, ground_truth))
```

```
In [13]: print('User-based CF RMSE: ' + str(rmse(user_prediction, test_data_matrix)))
         print('Item-based CF RMSE: ' + str(rmse(item_prediction, test_data_matrix)))
```

```
User-based CF RMSE: 3.1213177574547295
Item-based CF RMSE: 3.4493690282409926
```