



Text Analytics & Business Application

Netflix Competition

Qinglai He

Department of Operations and Information Management

Wisconsin School of Business

Who's watching?



Scarlett



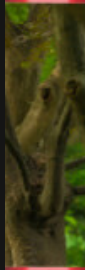
Sunny



Robin

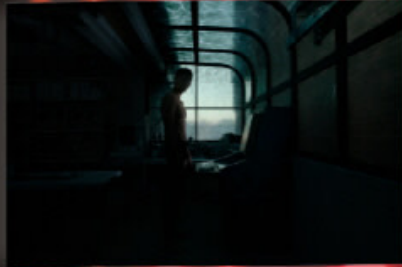


Dusty



NETFLIX

2022






WIKIPEDIA
The Free Encyclopedia

[Main page](#)
[Contents](#)
[Current events](#)
[Random article](#)
[About Wikipedia](#)
[Contact us](#)
[Donate](#)

[Contribute](#)

[Help](#)

 Not logged in

Article

[Talk](#)

Read

[Edit](#)

[View history](#)

Netflix Prize

From Wikipedia, the free encyclopedia

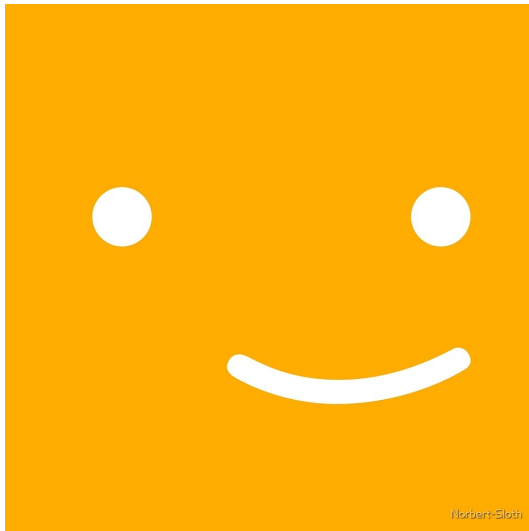
The **Netflix Prize** was an open competition for the best [collaborative filtering algorithm](#) to predict user ratings for [films](#), based on previous ratings without any other information about the users or films, i.e. without the users being identified except by numbers assigned for the contest.

The competition was held by [Netflix](#), an online DVD-rental and video streaming service, and was open to anyone who is neither connected with Netflix (current and former employees, agents, close relatives of Netflix employees, etc.) nor a resident of certain blocked countries (such as Cuba or North Korea).^[1] On September 21, 2009, the grand prize of US\$1,000,000 was given to the BellKor's Pragmatic Chaos team which bested Netflix's own algorithm for predicting ratings by 10.06%.^[2]



What Would be Dan's Rating on The Godfather?

What are the deciding factors?

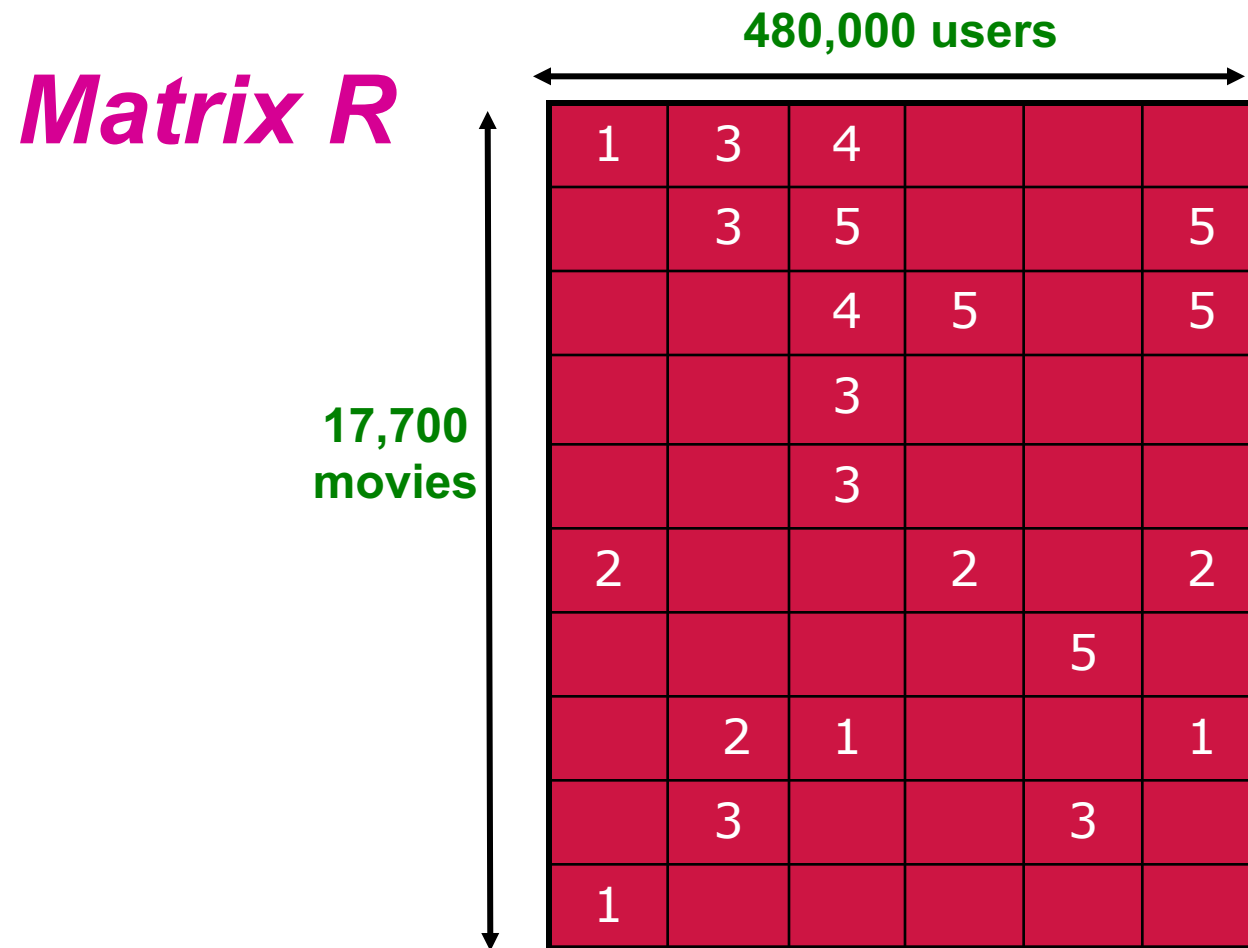


The Netflix Prize

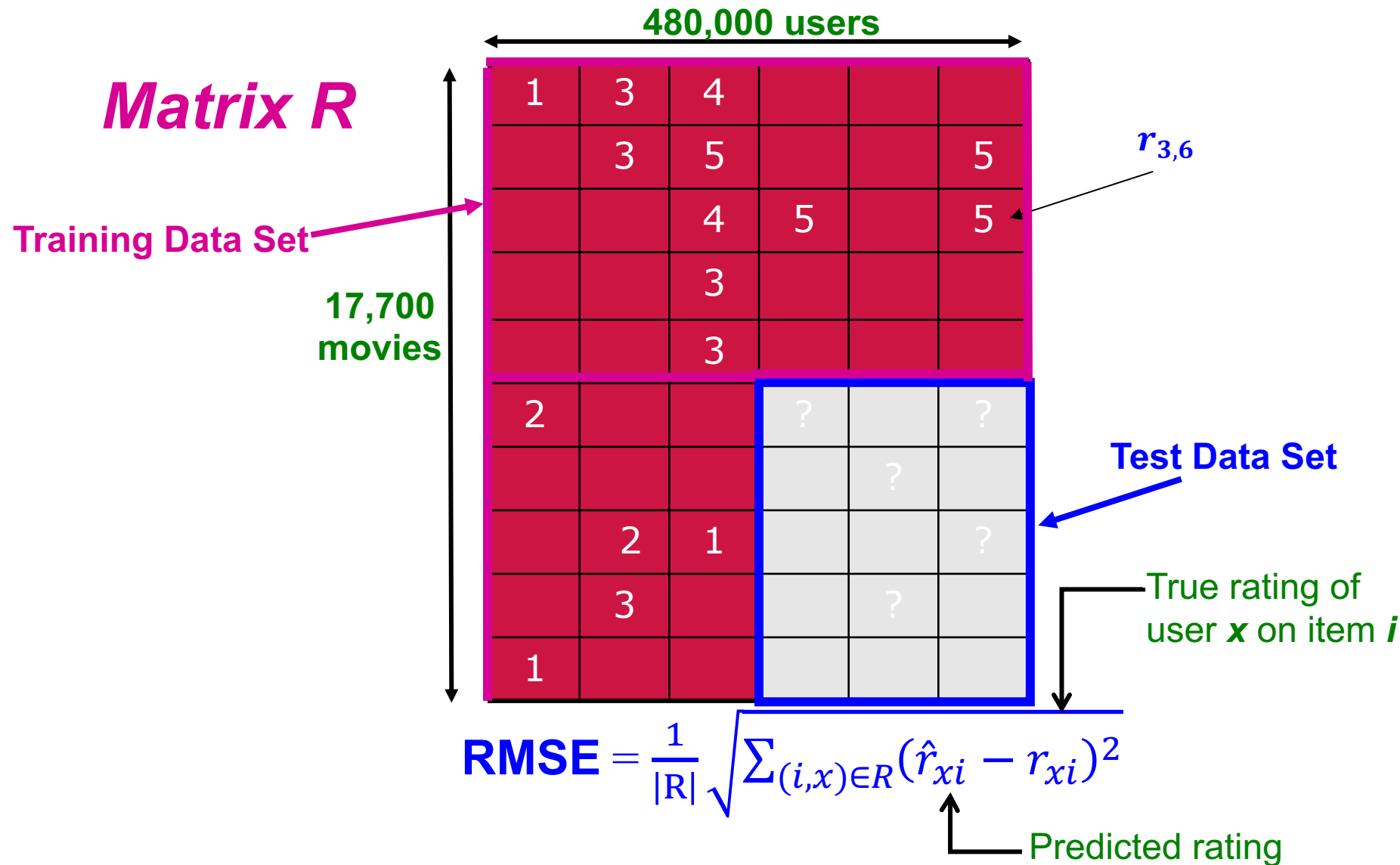
- Training data
 - 100 million ratings, 480,000 users, 17,770 movies
 - 6 years of data: 2000-2005
- Test data
 - 2.8 million user ratings
 - Test set (1,408,789 ratings), used to determine winners
 - Quiz set (1,408,342 ratings), used to calculate leaderboard scores
 - Evaluation criterion: Root Mean Square Error (RMSE) = $\frac{1}{|R|} \sqrt{\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2}$
 - Netflix's system RMSE: 0.9514 (on the quiz set)
- Competition
 - 2,700+ teams
 - \$1 million prize for 10% improvement on Netflix



The Netflix Utility Matrix R

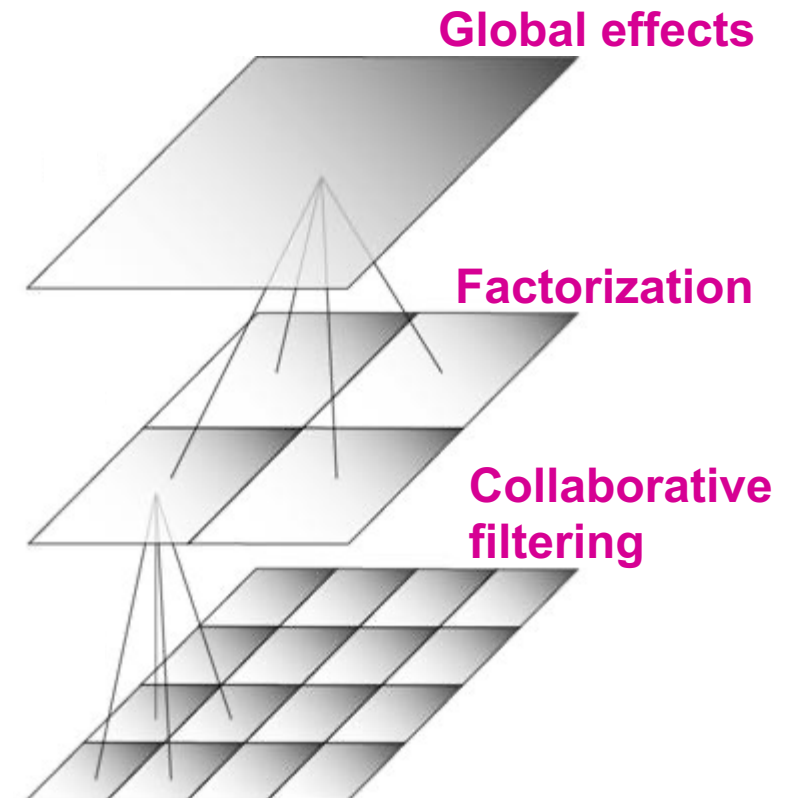


Utility Matrix R: Evaluation



BellKor Recommender System

- The winner of the Netflix Challenge!
- Multi-scale modeling of the data:
Combine top level, “regional” modeling of the data, with a refined, local view:
 - Global:
 - Overall deviations of users/movies
 - Factorization:
 - Addressing “regional” effects
 - Collaborative filtering:
 - Extract local patterns



Modeling Local & Global Effects

- Global:
 - 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
- Local neighborhood (CF/NN):
 - *Joe* didn't like related movie *Signs*
 - ⇒ Final estimate:
Joe will rate The Sixth Sense 3.8 stars



Recap: Collaborative Filtering (CF)

- Earliest and most popular collaborative filtering method
- Derive unknown ratings from those of “similar” movies (item-item variant)
- Define similarity measure s_{ij} of items i and j
- Select k -nearest neighbors, compute the rating
 - $N(i; x)$: items most similar to i that were rated by x

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

s_{ij} ... similarity of items i and j
 r_{xj} ... rating of user x on item j
 $N(i; x)$... set of items similar to item i that were rated by x



Modeling Local & Global Effects

- In practice we get better estimates if we model deviations:

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

baseline estimate for r_{xi}

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

μ = overall mean rating

b_x = rating deviation of user x
= (avg. rating of user x) $- \mu$

b_i = (avg. rating of movie i) $- \mu$

Problems/Issues:

- 1) Similarity measures are “arbitrary”
- 2) Pairwise similarities neglect interdependencies among users
- 3) Taking a weighted average can be restricting

Solution: Instead of s_{ij} use w_{ij} that we estimate directly from data



Idea: Interpolation Weights w_{ij}

- Use a weighted sum rather than weighted avg.:

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

- A few notes:
 - $N(i; x)$... set of movies rated by user x that are similar to movie i
 - w_{ij} is the interpolation weight (some real number)
 - We allow: $\sum_{j \in N(i,x)} w_{ij} \neq 1$
 - w_{ij} models interaction between pairs of movies (it does not depend on user x)



Idea: Interpolation Weights w_{ij}

- $\hat{r}_{xi} = b_{xi} + \sum_{j \in N(i,x)} w_{ij}(r_{xj} - b_{xj})$
- How to set w_{ij} ?
 - Remember, error metric is: $\frac{1}{|R|} \sqrt{\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2}$ or equivalently SSE: $\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2$
 - Find w_{ij} that minimize SSE on training data!
 - Models relationships between item i and its neighbors j
 - w_{ij} can be learned/estimated based on x and all other users that rated i



Recommendations via Optimization

- Goal: Make good recommendations
 - Quantify goodness using RMSE:
Lower RMSE \Rightarrow better recommendations
 - Want to make good recommendations on items that user has not yet seen.
- Let's set build a system such that it works well on known (user, item) ratings
 - And hope the system will also predict well the unknown ratings

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



Recommendations via Optimization

- Idea: Let's set values w such that they work well on known (user, item) ratings
- How to find such values w ?
- Idea: Define an objective function and solve the optimization problem
- Find w_{ij} that minimize SSE on training data!

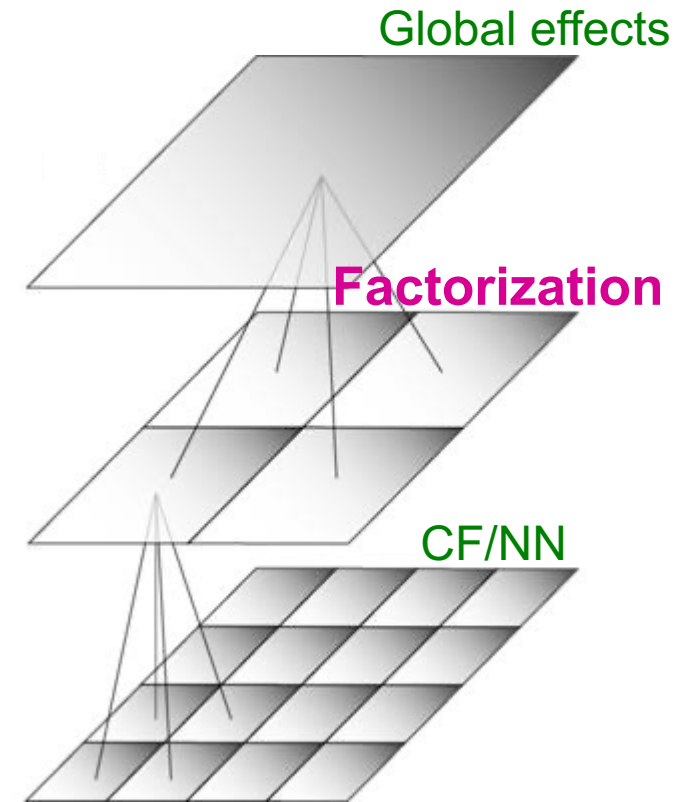
$$J(w) = \sum_{x,i} \left(\underbrace{\left[b_{xi} + \sum_{j \in N(i;x)} w_{ij}(r_{xj} - b_{xj}) \right]}_{\text{Predicted rating}} - \underbrace{r_{xi}}_{\text{True rating}} \right)^2$$

- Think of w as a vector of numbers

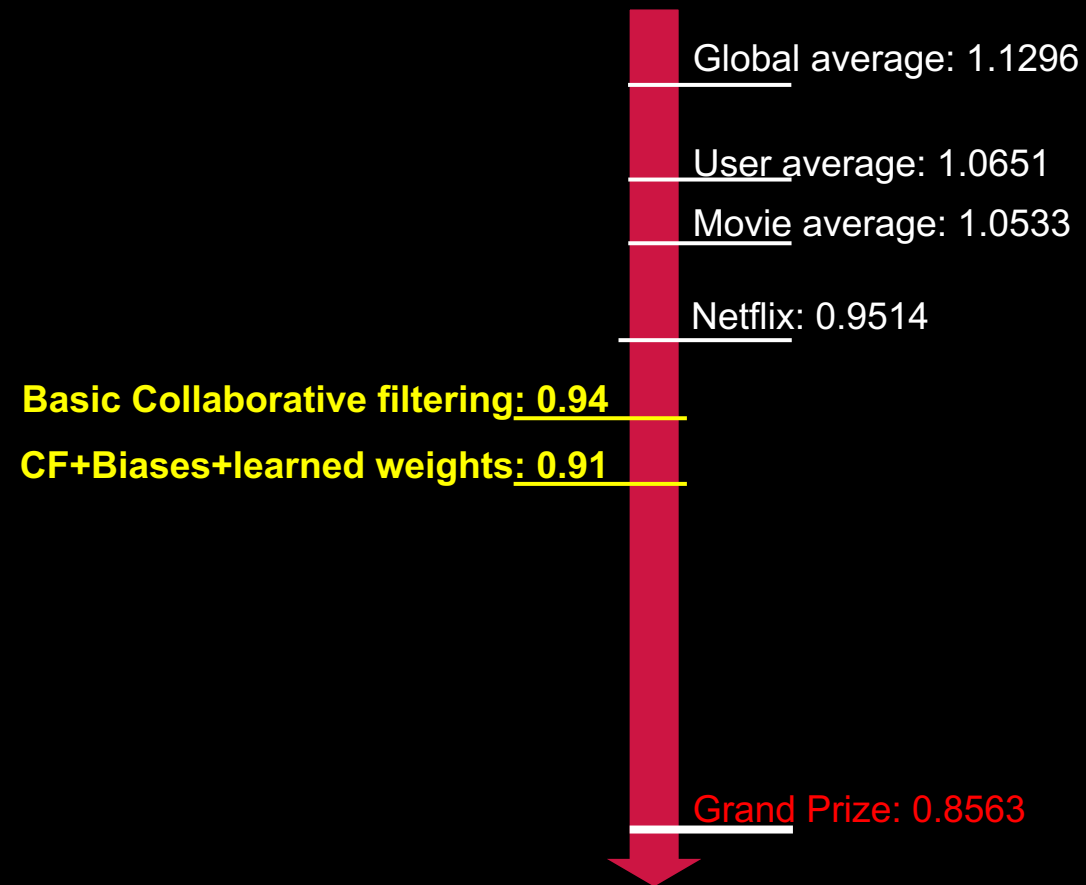


Interpolation Weights

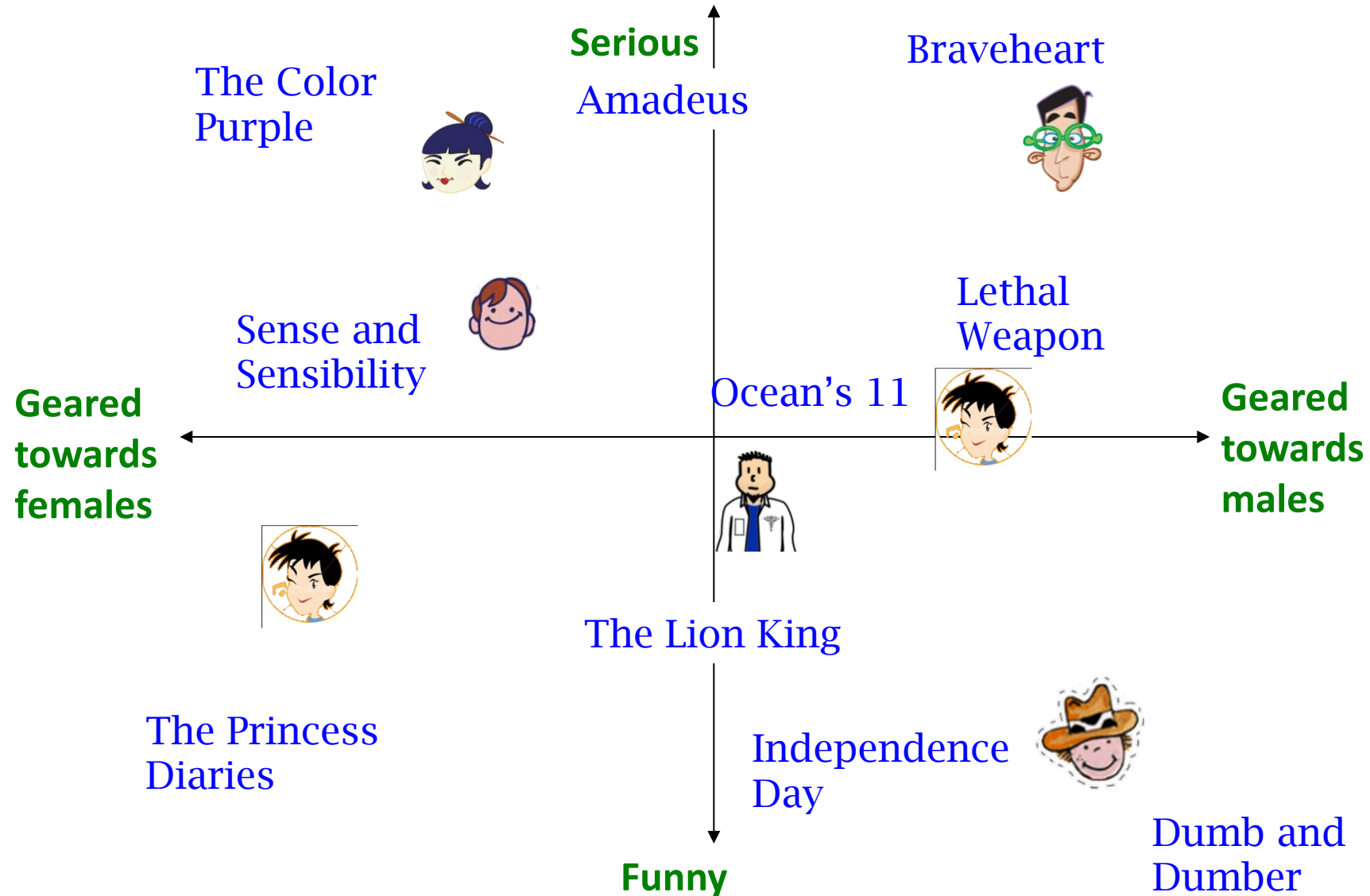
- So far: $\widehat{r}_{xi} = b_{xi} + \sum_{j \in N(i;x)} w_{ij}(r_{xj} - b_{xj})$
 - Weights w_{ij} derived based on their role; **no use of an arbitrary similarity measure** ($w_{ij} \neq s_{ij}$)
 - Explicitly account for interrelationships among the neighboring movies
- Next: Latent factor model
 - Extract “regional” correlations



Performance of Various Methods

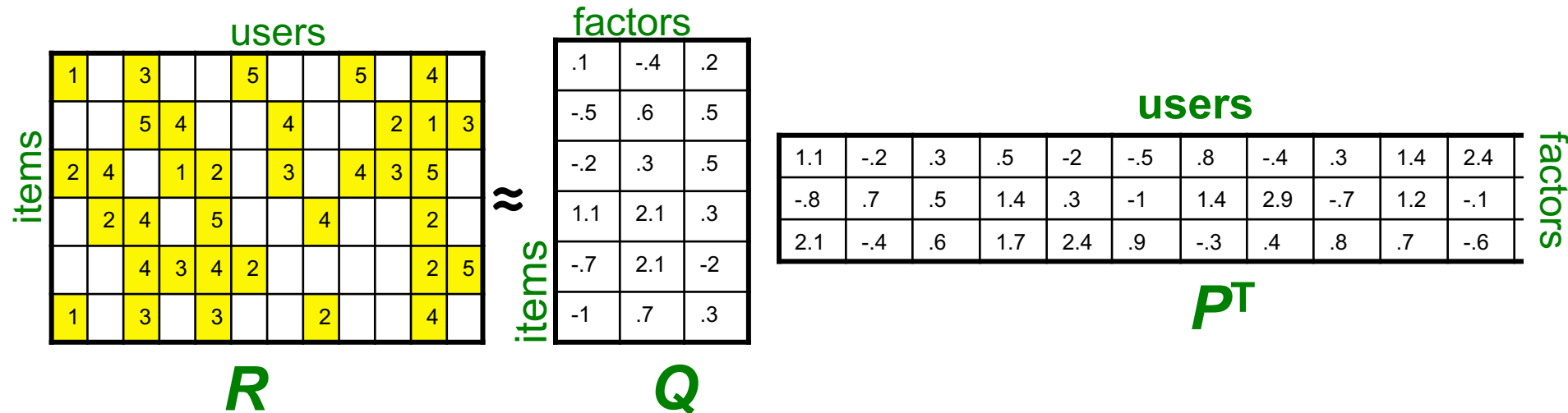


Latent Factor Models



Latent Factor Models

- Singular-value decomposition (SVD) on Netflix data: $R \approx Q \cdot P^T$ **SVD:** $A = U \Sigma V^T$



- For now let's assume we can approximate the rating matrix R as a product of "thin" $Q \cdot P^T$
 - R has missing entries but let's ignore that for now!
 - Basically, we will want the reconstruction error to be small on known ratings and we don't care about the values on the missing ones



Ratings as Products of Factors

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

[illegible]

$$\begin{aligned}\hat{r}_{xi} &= \mathbf{q}_i \cdot \mathbf{p}_x \\ &= \sum_f \mathbf{q}_{if} \cdot \mathbf{p}_{xf}\end{aligned}$$

\mathbf{q}_i = row i of \mathbf{Q}
 \mathbf{p}_x = column x of \mathbf{P}^\top

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

Q

factors

	users											
	1.1	-.2	.3	.5	-.2	-.5	.8	-.4	.3	1.4	2.4	-.9
factors	-.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
	PT											

PT



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

items

factors

Q

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

factors

users

P^T

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

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	2.4	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

items

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

f factors

Q

f factors

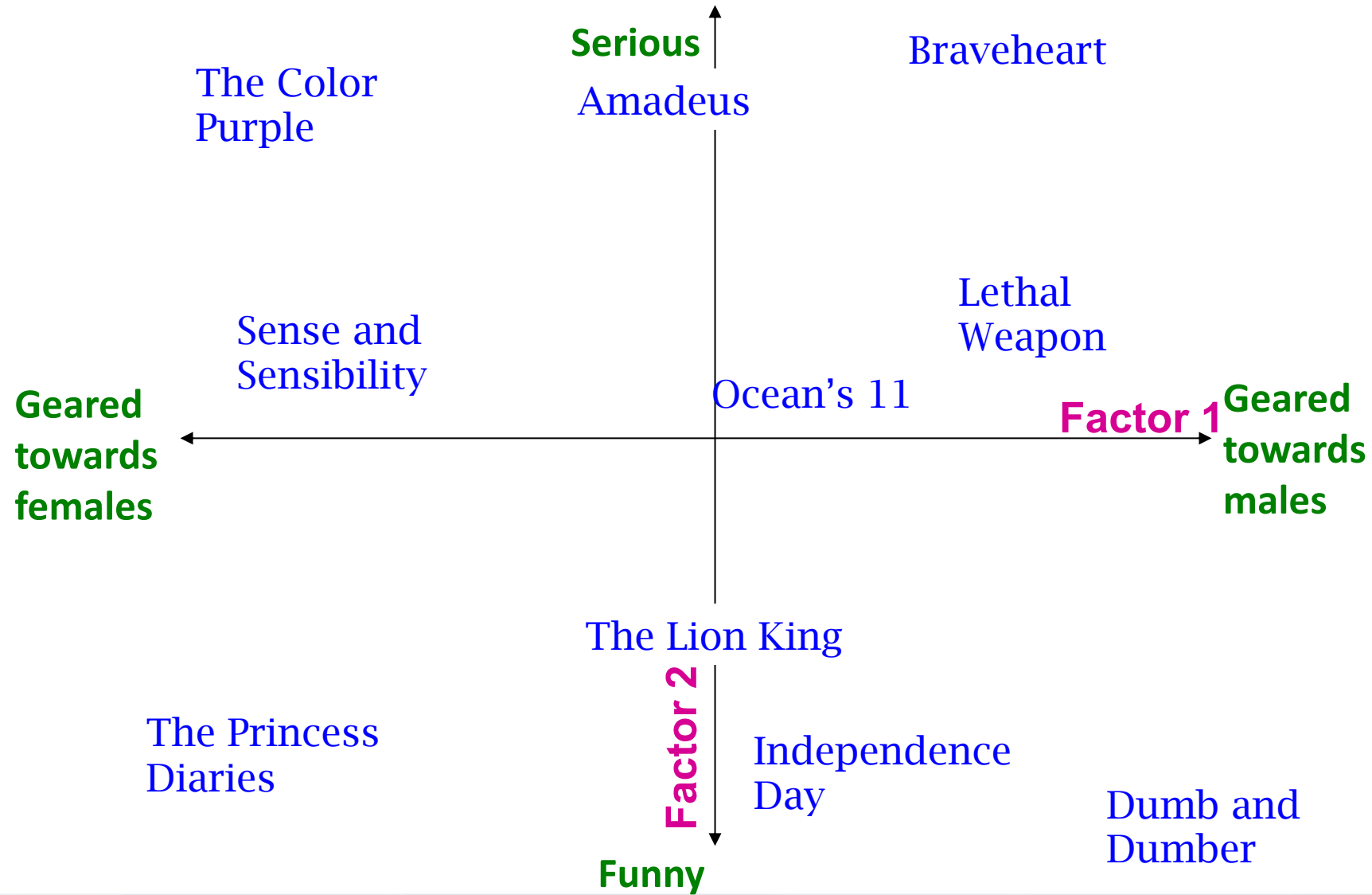
users

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

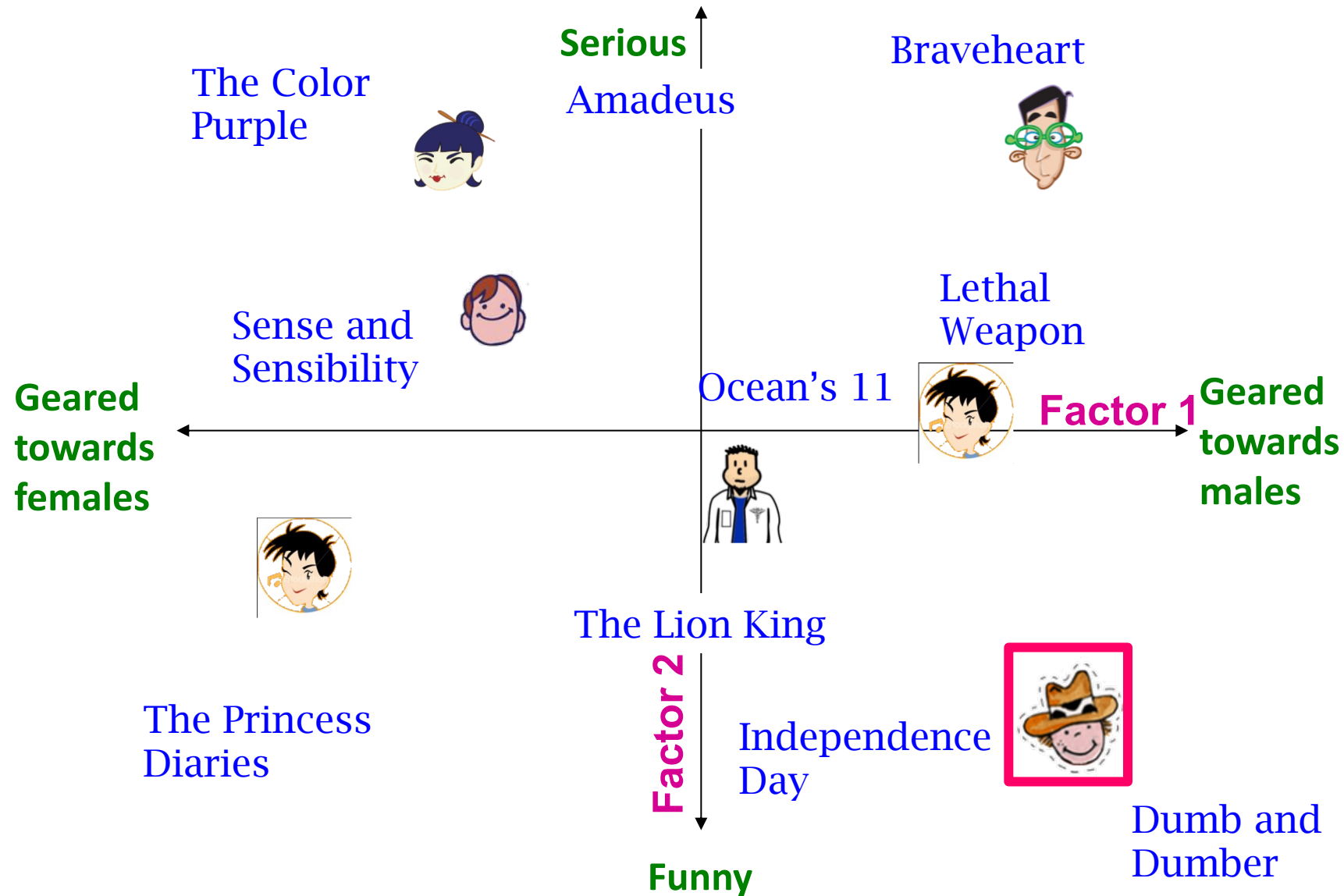
P^T

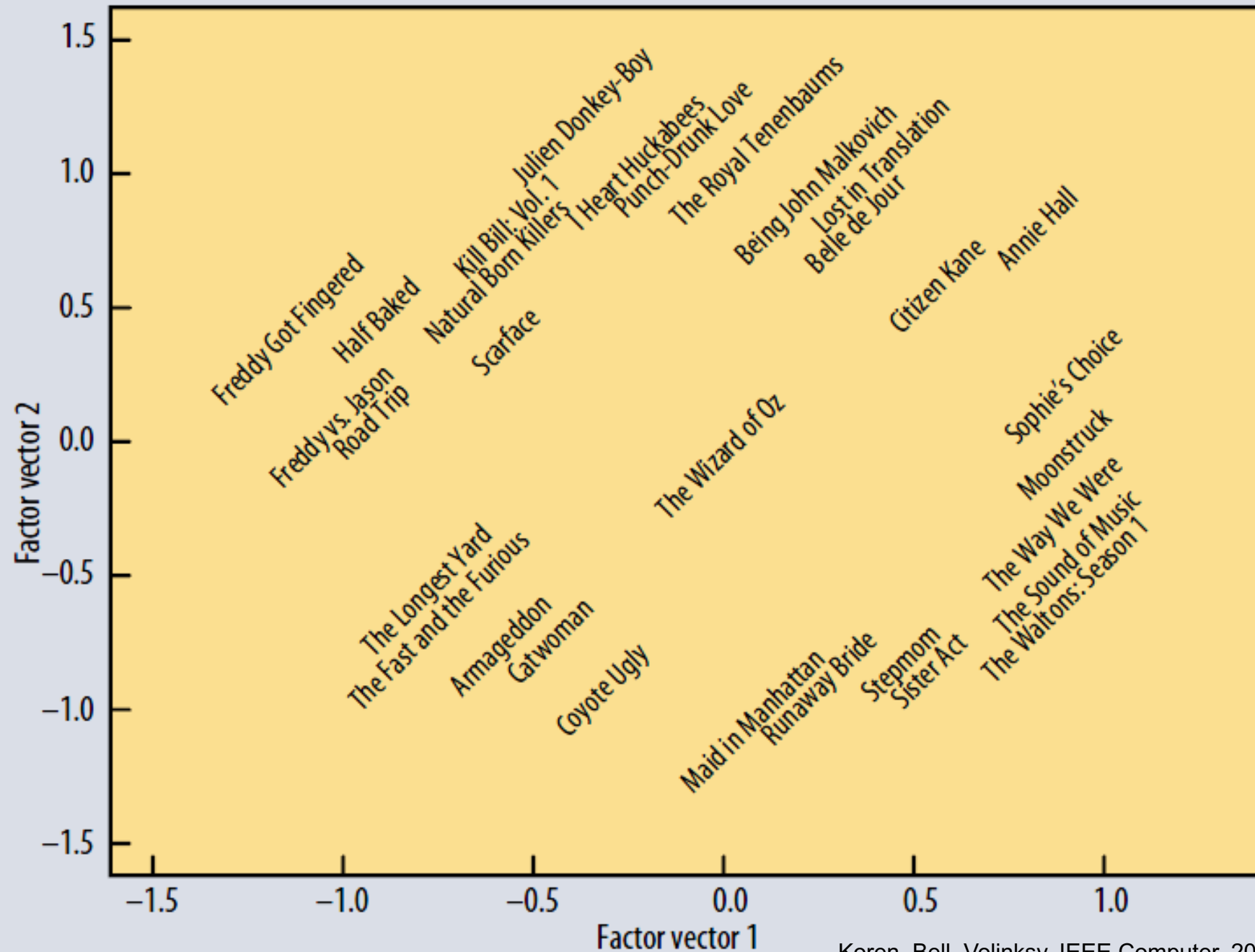


Latent Factor Models



Latent Factor Models

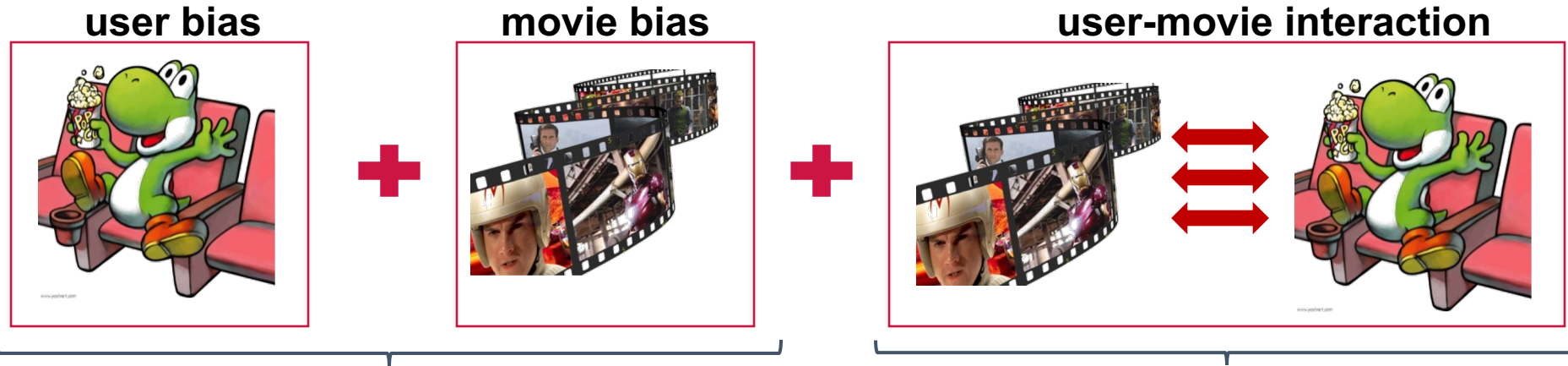




Extending Latent Factor Model to Include Biases



Modeling Biases and Interactions



Baseline predictor

- Separates users and movies
- Benefits from insights into user's behavior
- Among the main practical contributions of the competition

User-Movie interaction

- Characterizes the matching between users and movies
- Attracts most research in the field
- Benefits from algorithmic and mathematical innovations

- μ = overall mean rating
- b_x = bias of user x
- b_i = bias of movie i

Baseline Predictor

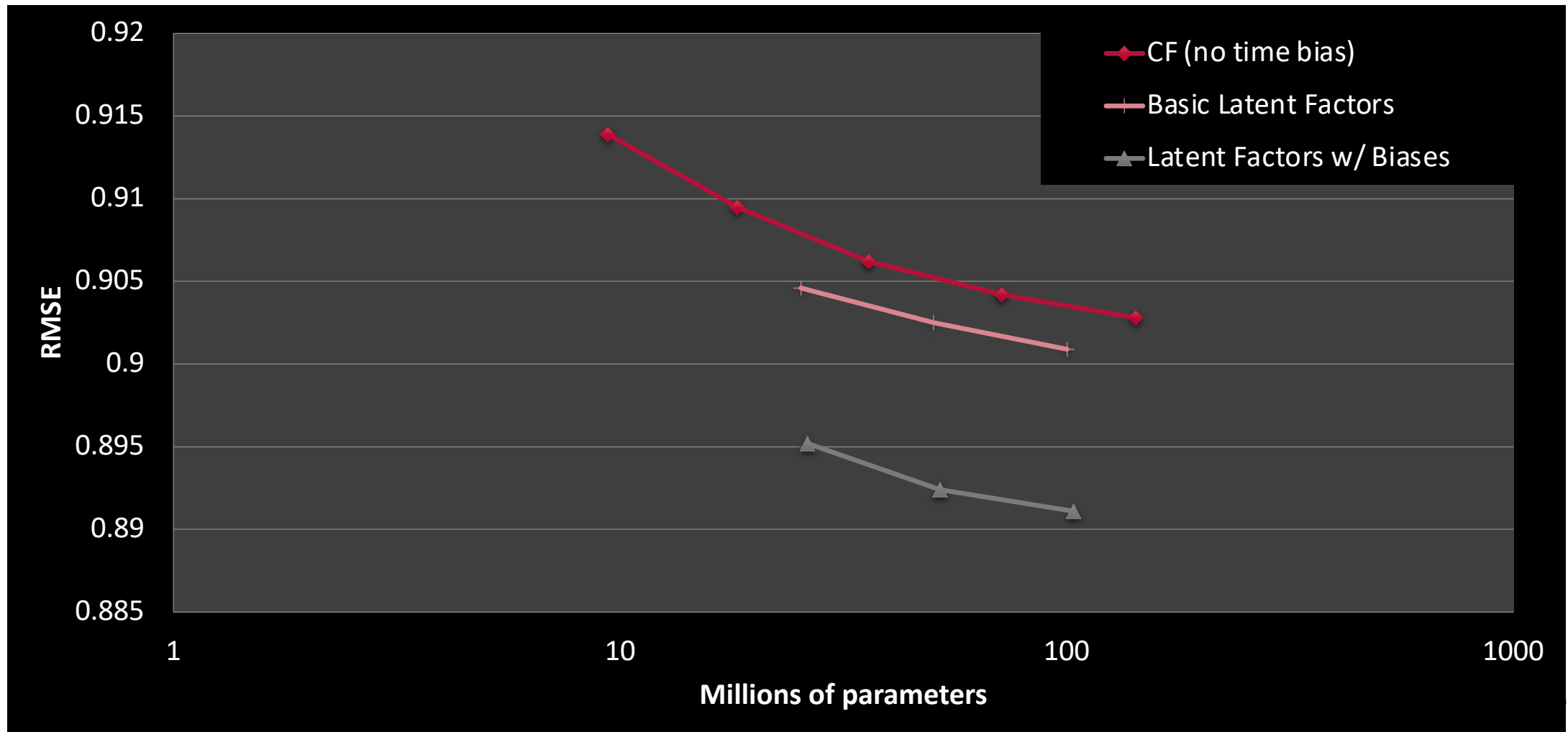
- We have expectations on the rating by user x of movie i , even without estimating x 's attitude towards movies like i



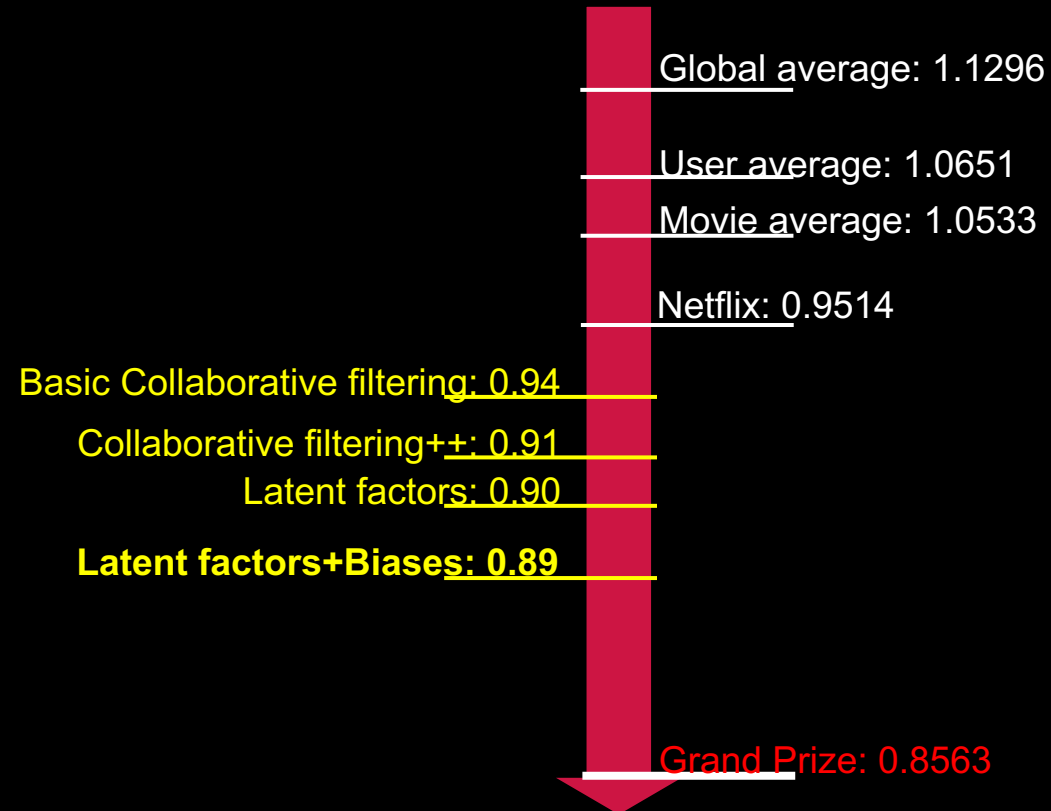
- Rating scale of user x
- Values of other ratings user gave recently (day-specific mood, anchoring, multi-user accounts)

- (Recent) popularity of movie i
- Selection bias; related to number of ratings user gave on the same day ("frequency")

Performance of Various Methods



Performance of Various Methods

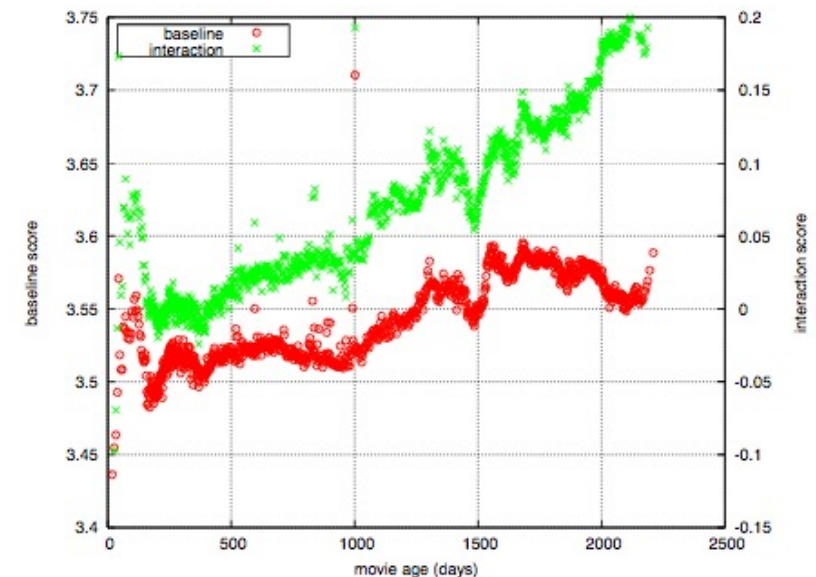
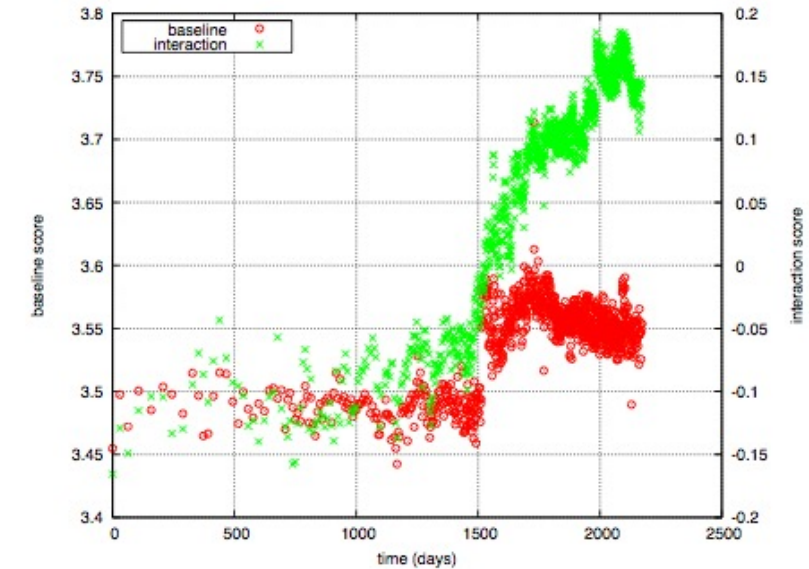


The Netflix Challenge: 2006-09



Temporal Biases Of Users

- Sudden rise in the average movie rating (early 2004)
 - Improvements in Netflix UI
 - Meaning of rating changed
- Movie age
 - Users prefer new movies without any reasons
 - Older movies are just inherently better than newer ones



Temporal Biases & Factors

- Original model:

$$r_{xi} = m + b_x + b_i + q_i \cdot p_x$$

- Add time dependence to biases:

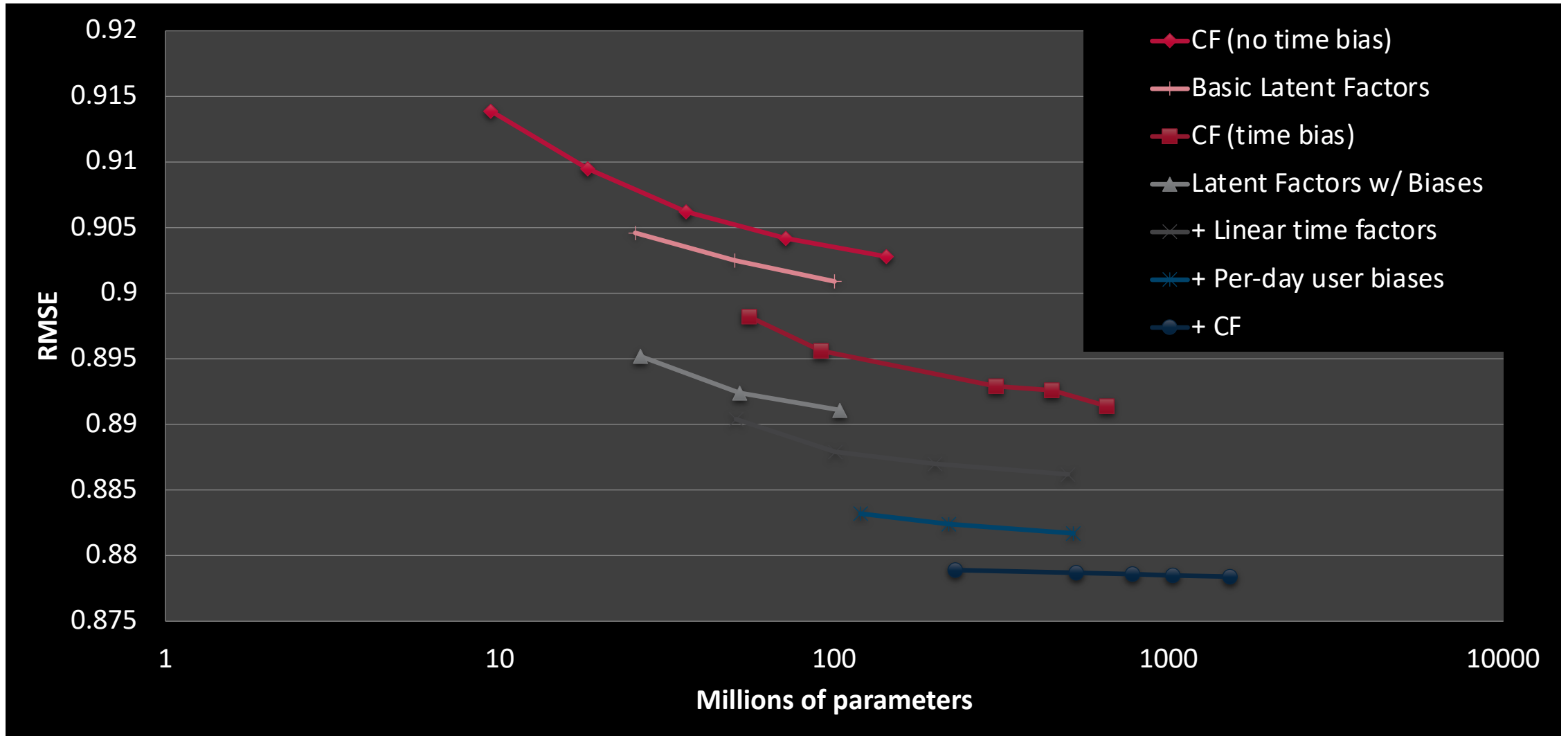
$$r_{xi} = m + b_x(t) + b_i(t) + q_i \cdot p_x$$

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

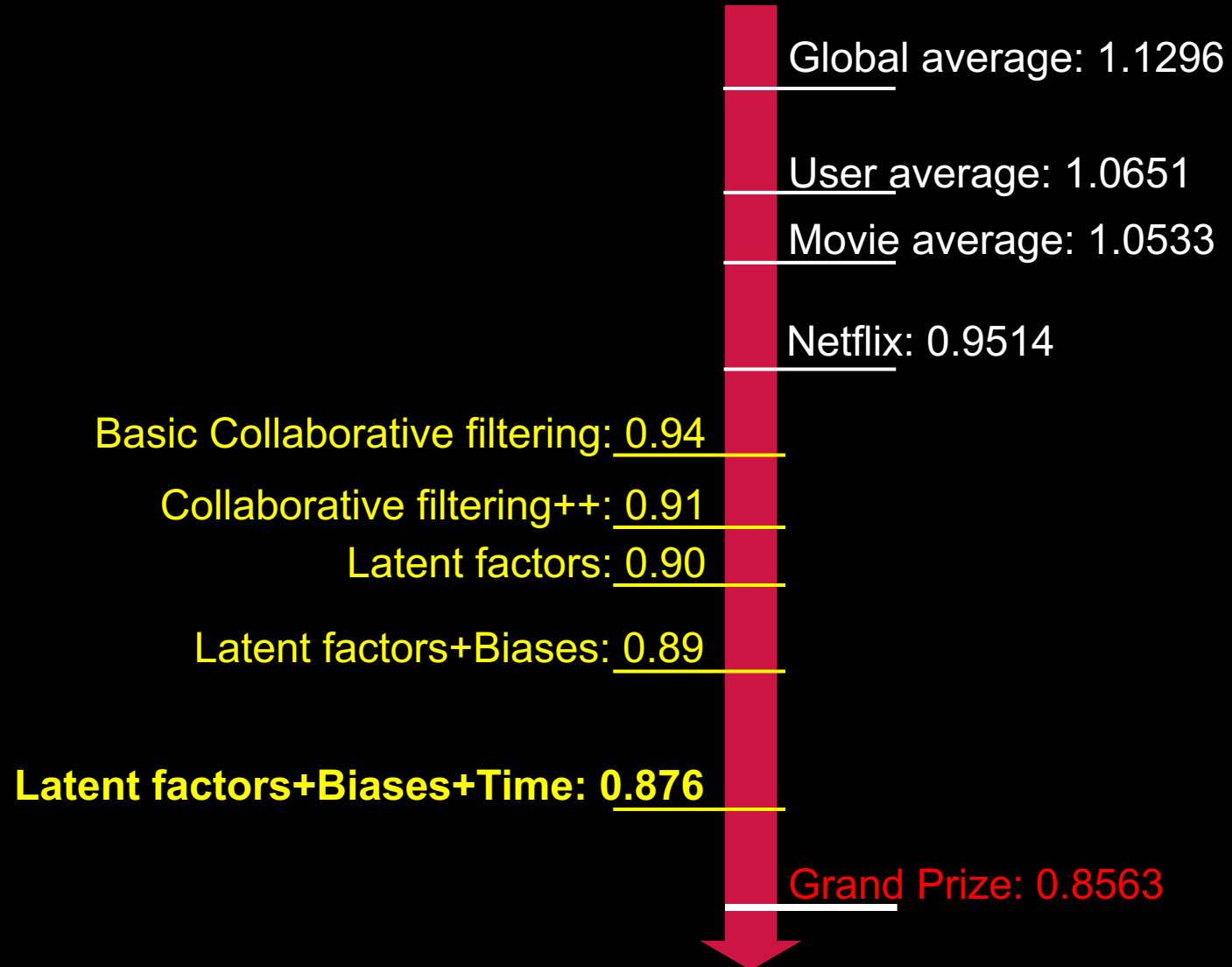
- Make parameters b_x and b_i to depend on time
 - (1) Parameterize time-dependence by linear trends
 - (2) Each bin corresponds to 10 consecutive weeks
- Add temporal dependence to factors
 - $p_x(t)$... user preference vector on day t



Adding Temporal Effects



Pe



Netflix Prize

COMPLETED

[Home](#) [Rules](#) [Leaderboard](#) [Update](#) [Download](#)

Leaderboard

Showing Test Score. [Click here to show quiz score](#)Display top 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	0.8582	9.89	2009-07-10 21:11:10
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

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets,
<http://www.mmds.org>



Million \$ Awarded Sept 21st 2009



Exercises using Google Colab

