Information Storage and Retrieval

CSCE 670
Texas A&M University
Department of Computer Science & Engineering
Instructor: Prof. James Caverlee

Recommenders
23 March 2017

Slides adapted from Jure Leskovec, Yehuda Koren, and others

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 squared error (RMSE)
  - Netflix Cinematch system RMSE: 0.9514
- Competition
  - 2700+ teams
  - $1 million grand prize for 10% improvement over Netflix

RMSE

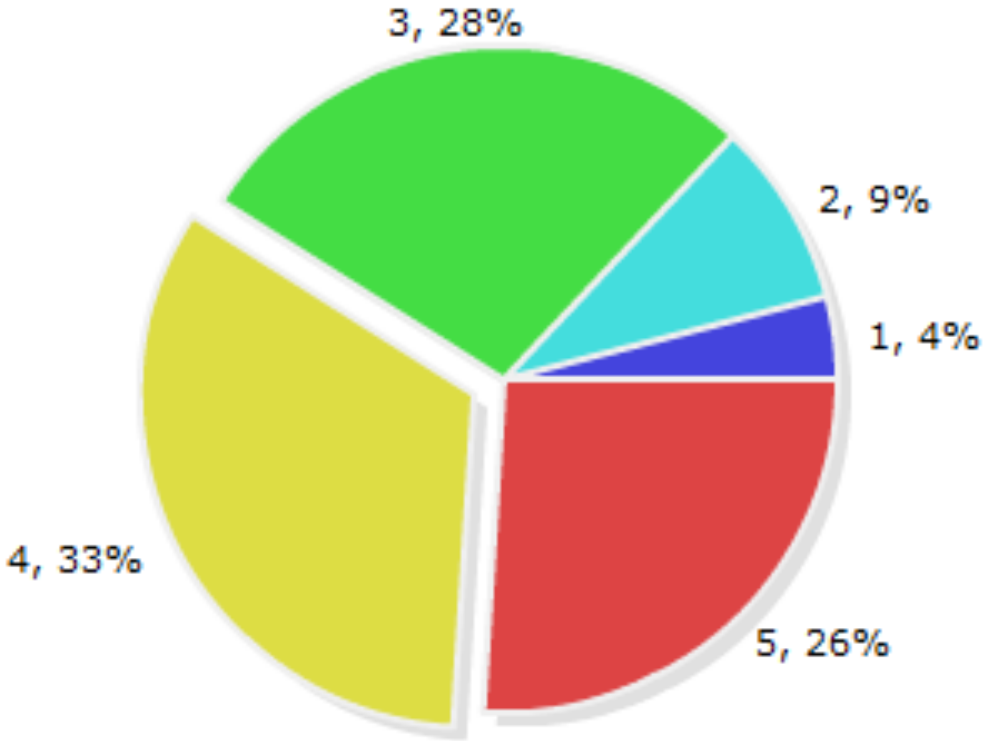
- Compare predictions with known ratings

- My system predicted you would rate
  - The Shawshank Redemption as **4.3 stars**
    - In reality, you gave it **5 stars**
  - The Matrix with **3.9 stars**
    - In reality, you gave it **4 stars**

- RMSE = sqrt(1/2 * (4.3 - 5)^2 + (3.9 - 4)^2))

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

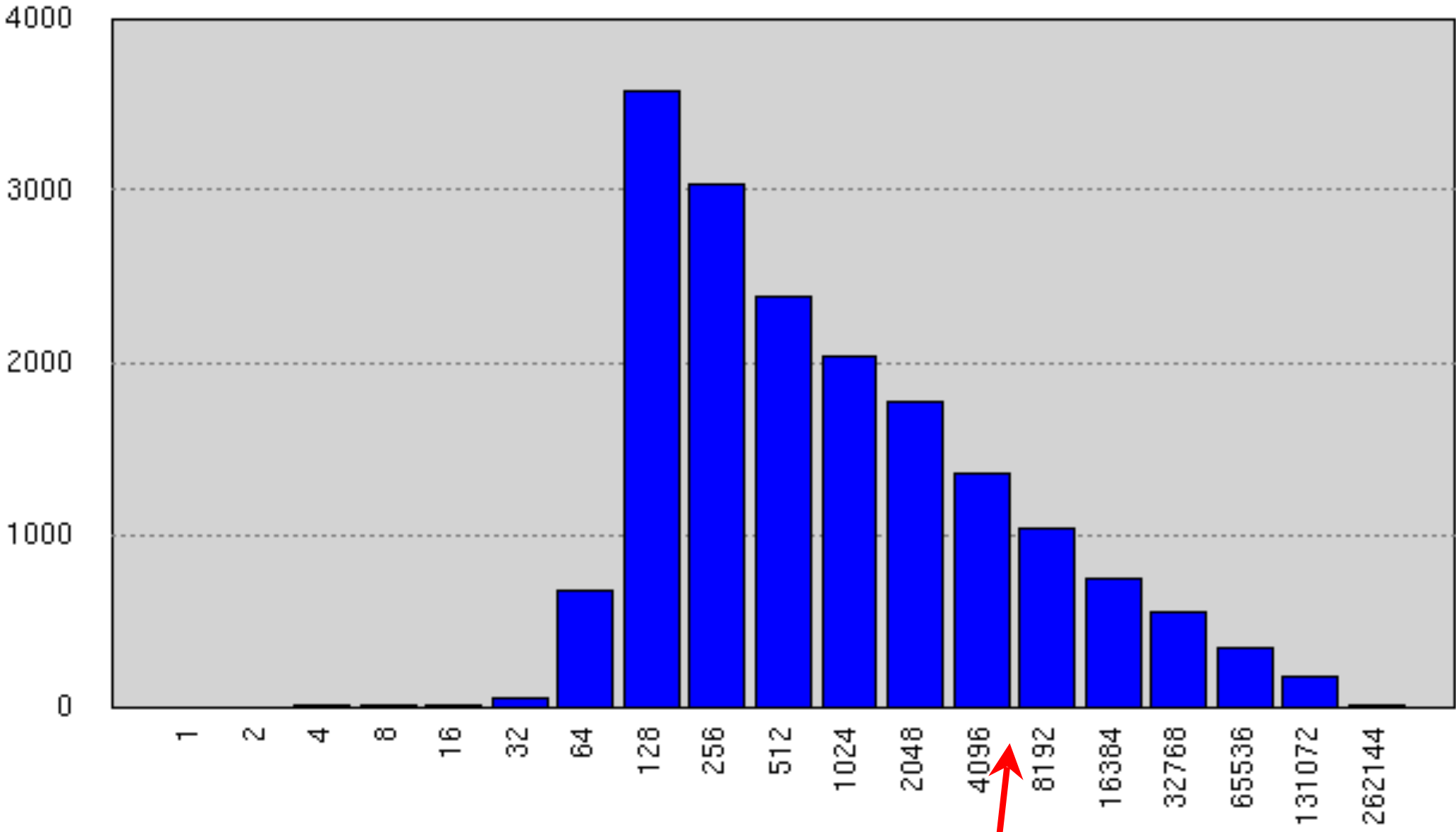
Overall rating distribution



- Third of ratings are 4s
- Average rating is 3.68

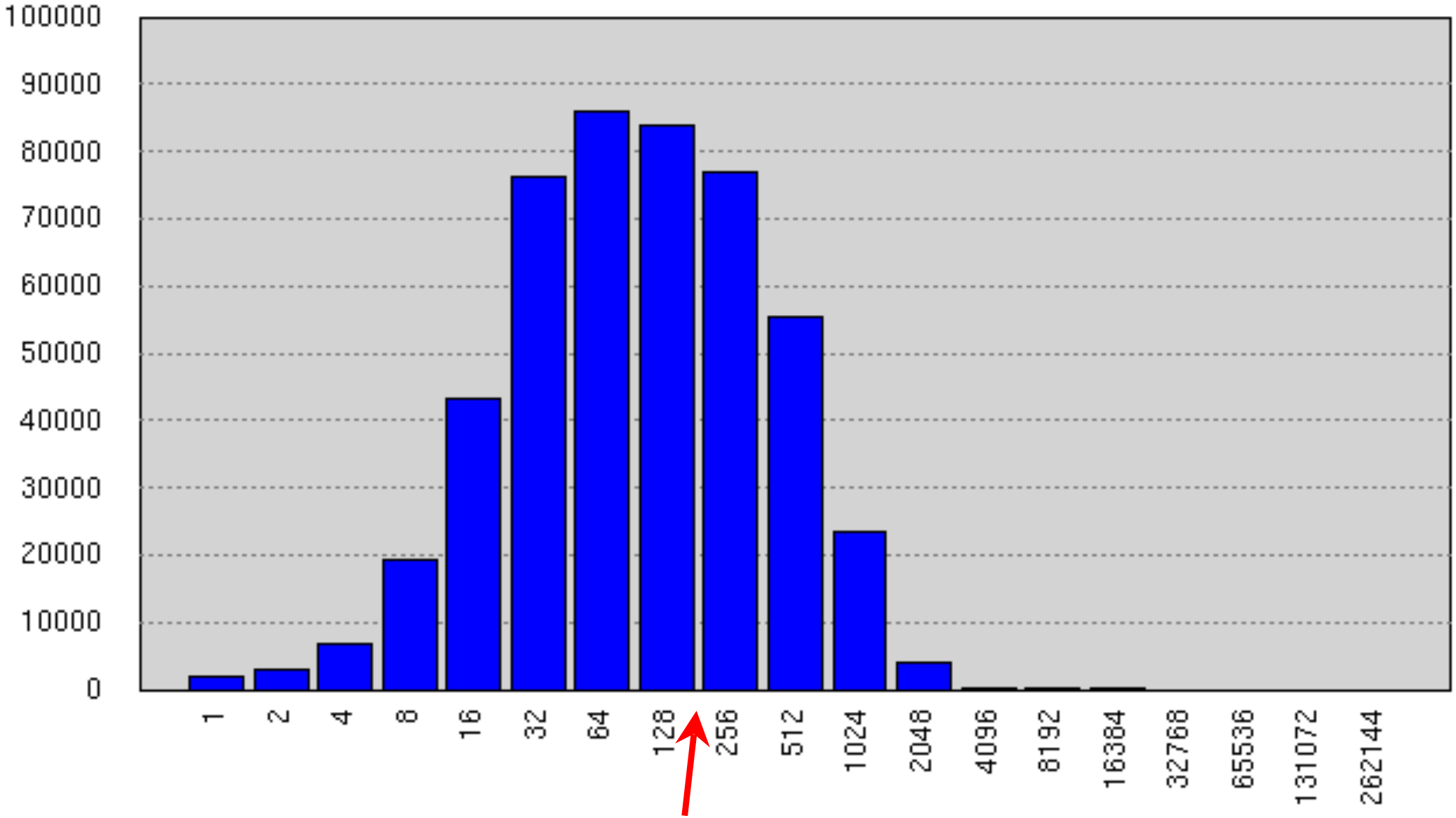
From TimelyDevelopment.com

#ratings per movie



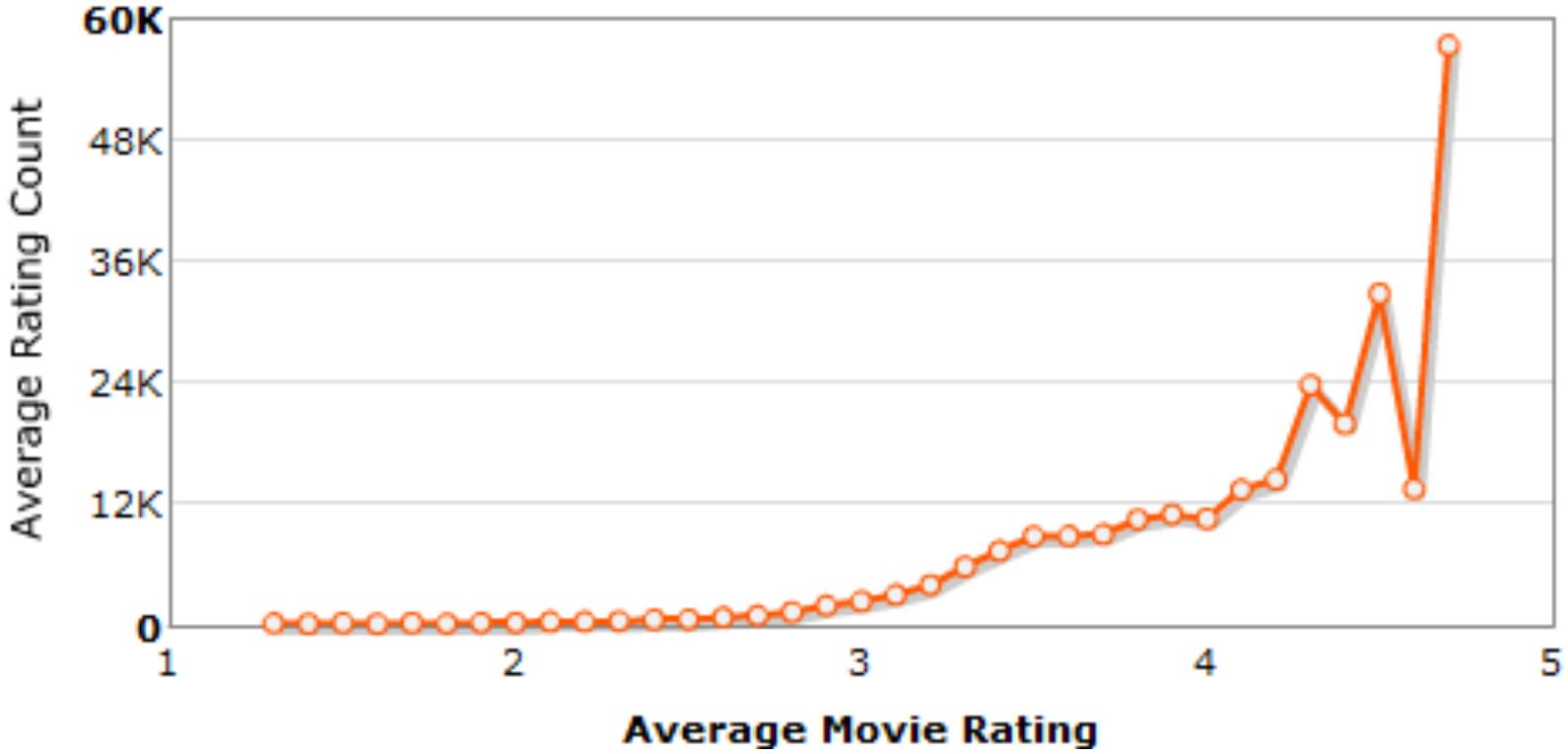
- Avg #ratings/movie: 5627

#ratings per user



- Avg #ratings/user: 208

Average movie rating by movie count



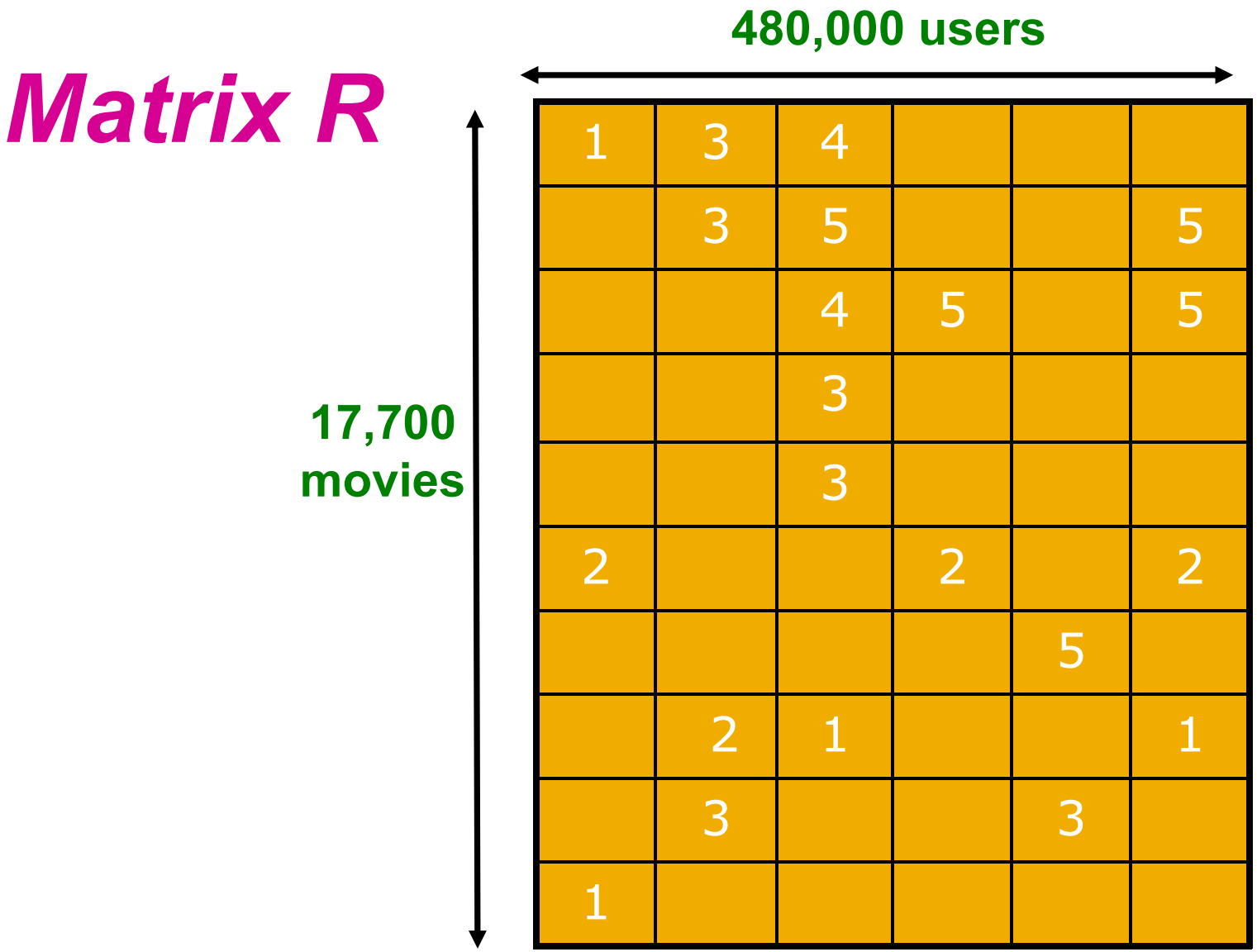
- More ratings to better movies

From TimelyDevelopment.com

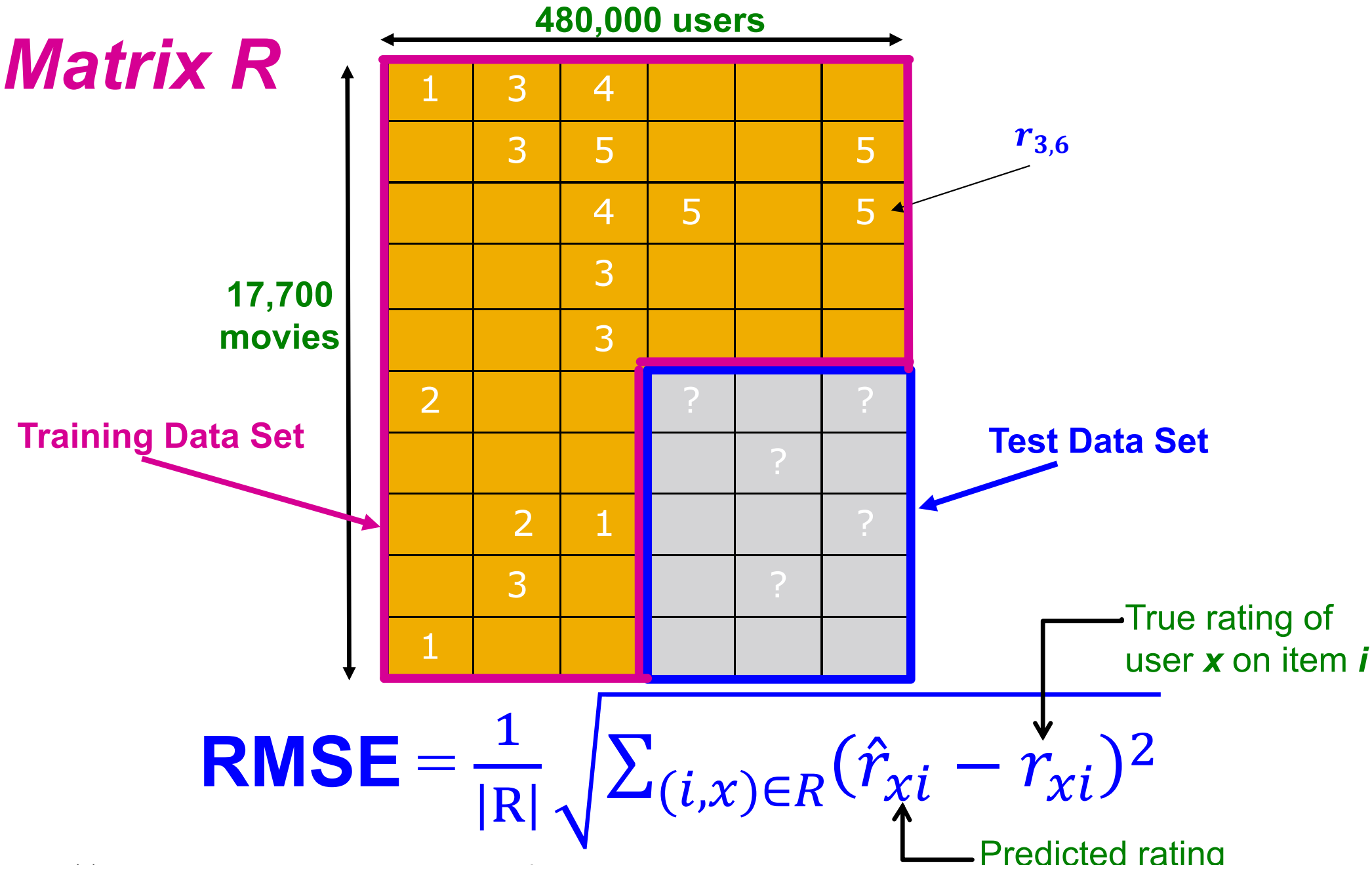
Most loved movies

Title	Avg rating	Count
The Shawshank Redemption	4.593	137812
Lord of the Rings: The Return of the King	4.545	133597
The Green Mile	4.306	180883
Lord of the Rings: The Two Towers	4.460	150676
Finding Nemo	4.415	139050
Raiders of the Lost Ark	4.504	117456
Forrest Gump	4.299	180736
Lord of the Rings: The Fellowship of the ring	4.433	147932
The Sixth Sense	4.325	149199
Indiana Jones and the Last Crusade	4.333	144027

The Netflix Utility Matrix R

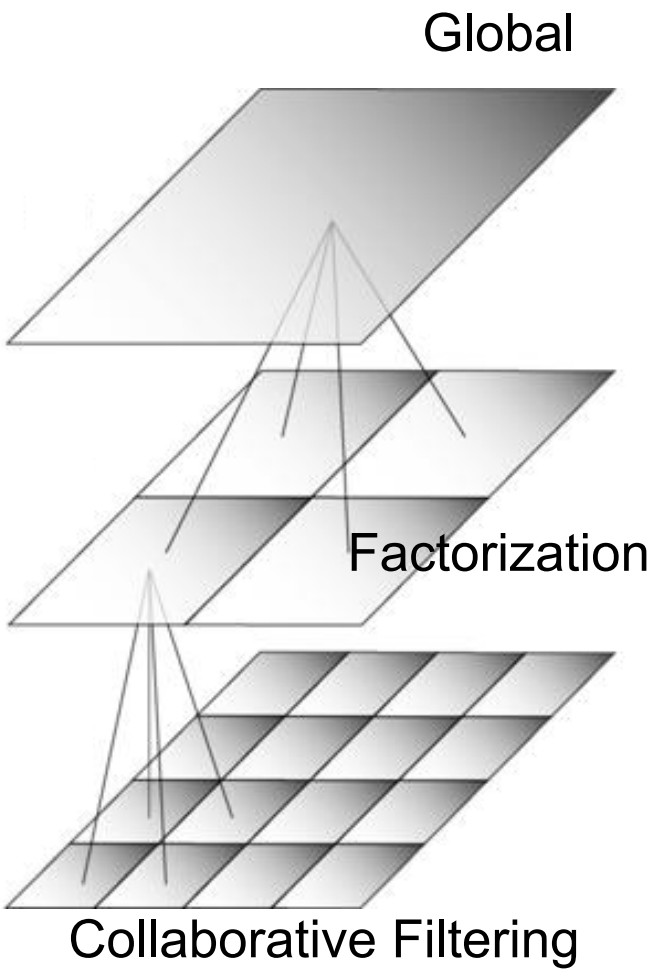


Utility Matrix R: Evaluation



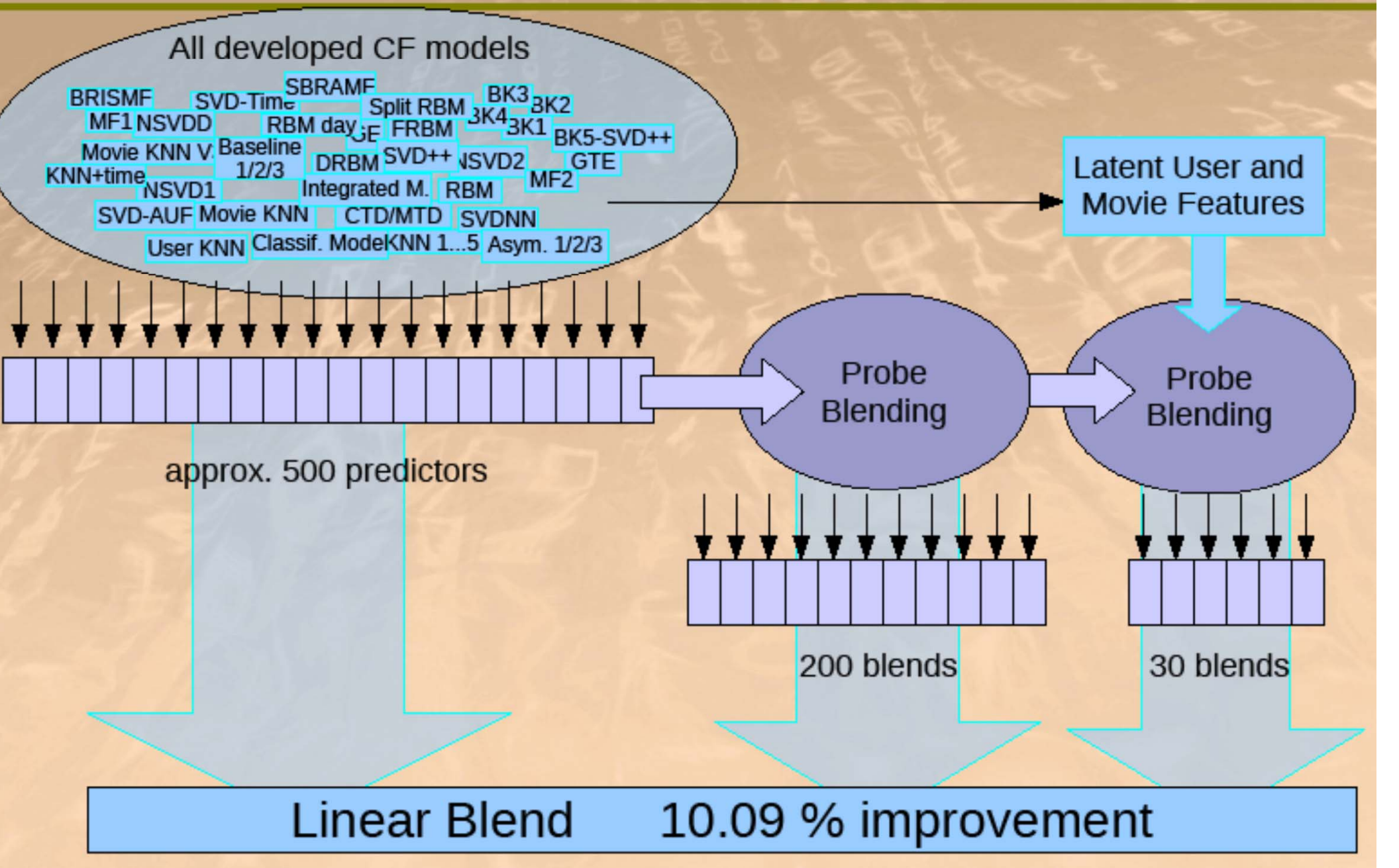
BellKor Recommender System

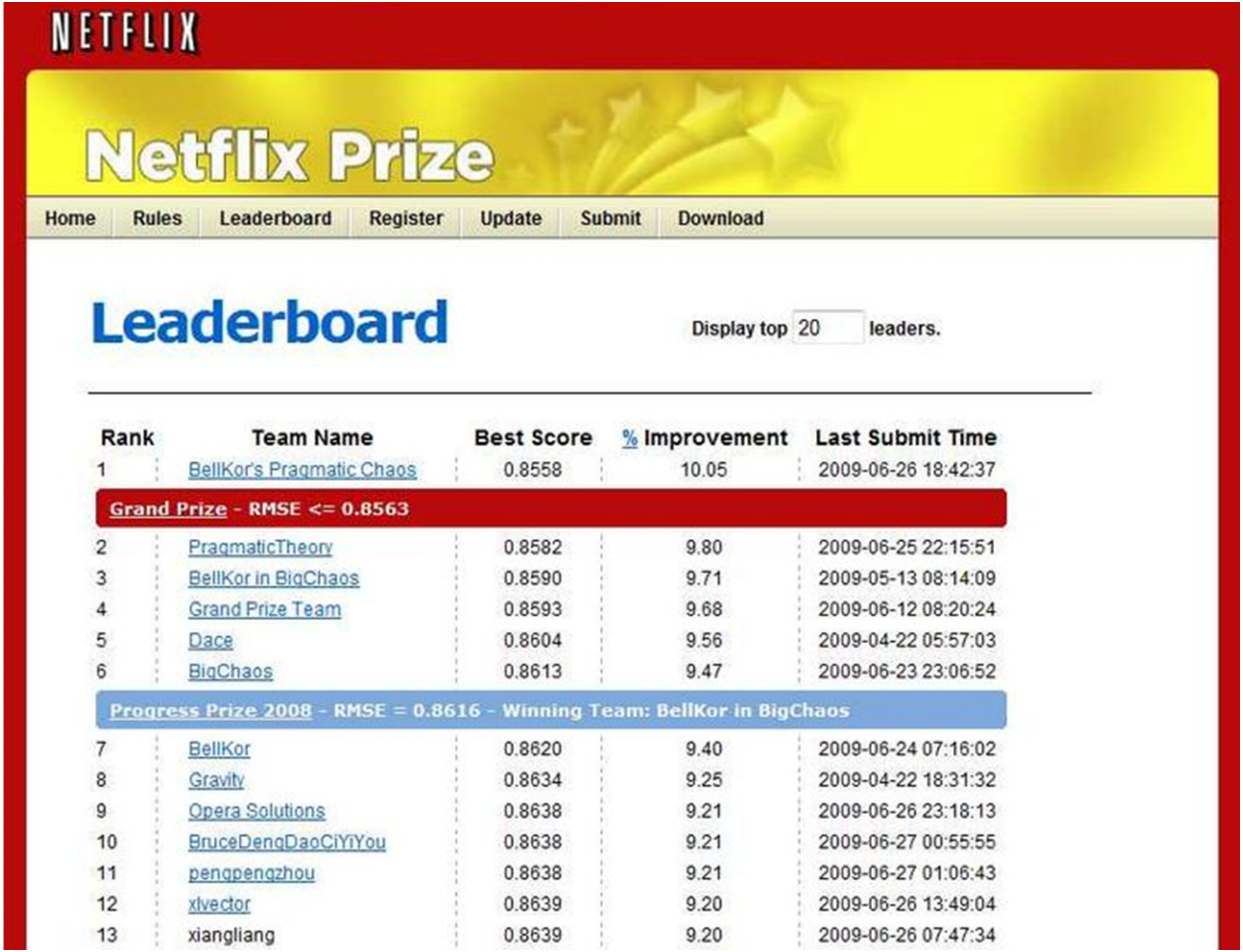
- The winner of the Netflix Challenge
- Multi-scale modeling of the data
  - **Global:**
    - Overall deviations of users/movies
  - **Factorization:**
    - Addressing "regional" effects
  - **Collaborative Filtering:**
    - Extract local patterns



The big picture

Solution of BellKor's Pragmatic Chaos

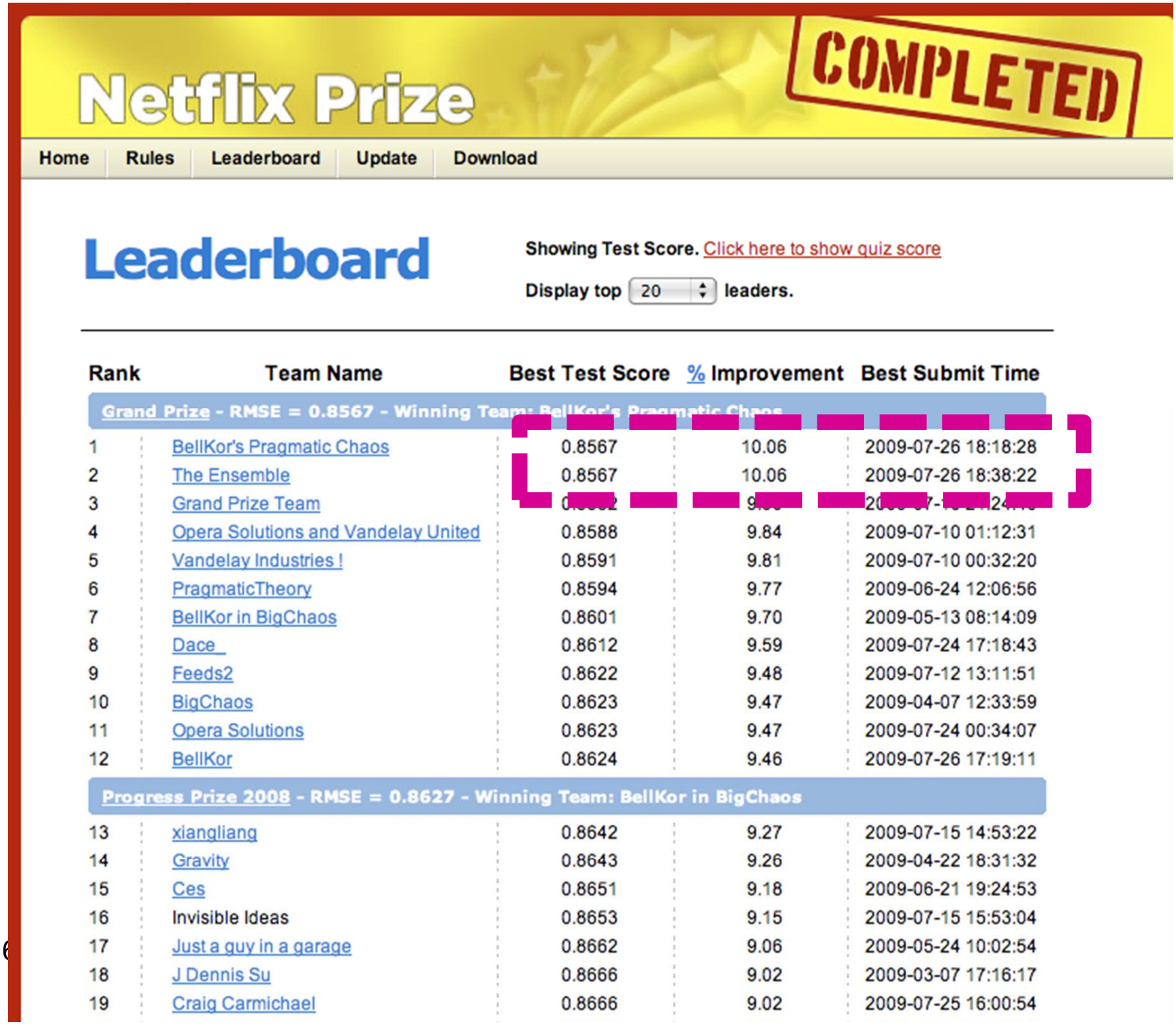




June 26th submission triggers 30-day "last call"

- **Ensemble team formed**
  - Group of other teams on leaderboard forms a new team
  - Relies on combining their models
  - Quickly also get a qualifying score over 10%
- **BellKor**
  - Continue to get small improvements in their scores
  - Realize that they are in direct competition with Ensemble
- **Strategy**
  - Both teams carefully monitoring the leaderboard
  - Only sure way to check for improvement is to submit a set of predictions
    - This alerts the other team of your latest score

- **Submissions limited to 1 a day**
  - Only 1 final submission could be made in the last 24h
- **24 hours before deadline…**
  - **BellKor** team member in Austria notices (by chance) that **Ensemble** posts a score that is slightly better than BellKor's
- **Frantic last 24 hours for both teams**
  - Much computer time on final optimization
  - Carefully calibrated to end about an hour before deadline
- **Final submissions**
  - **BellKor** submits a little early (on purpose), 40 mins before deadline
  - **Ensemble** submits their final entry 20 mins later
  - **….and everyone waits….**


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
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


2009
NETFLIX
DATE 09-21-09
PAY TO THE ORDER OF BellKor's Pragmatic Chaos
$1,000,000 00
AMOUNT ONE MILLION
00/100
FOR The Netflix Prize
Reed Hastings

Modeling Local and Global Effects

- **Global:**
  - Mean movie rating: **3.7 stars**
  - *The Sixth Sense* is **0.5** stars above average
  - Joe rates **0.2** stars below average
  - => Baseline estimate: Joe will rate *The Sixth Sense* **4 stars**
- **Local Neighborhood (Collaborative Filtering)**
  - Joe didn't like related movie *Signs*
  - =>Final estimate: Joe will rate *The Sixth Sense* **3.8 stars**

Recap: Collaborative Filtering

- **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 and 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) – μ
b_i = (*avg. rating of* movie i) – μ

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