Web-Scale Recommendation Systems

Yehuda Koren

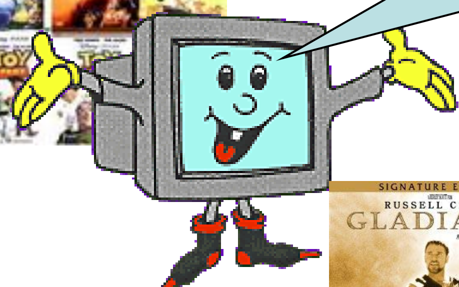
YAHOO! RESEARCH

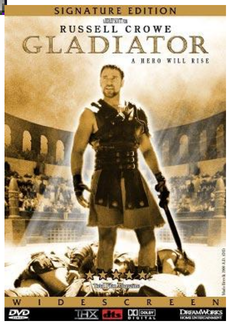
Talk outline

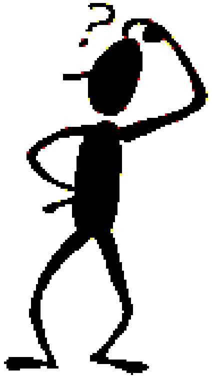
- Recommender systems – a quick intro
  - Neighborhood methods
  - Matrix factorization methods
- Biases and temporal dynamics
- Bootstrapping a recommender – ratings elicitation
- Y!Answers – combining multiple kinds of attributes and feedback
- Interpretation of user feedback: binary, numeric, or ordinal?
- Estimating confidence in recommendations
- KDD-Cup'2011

Recommender systems

We Know What You Ought To Be Watching This Summer





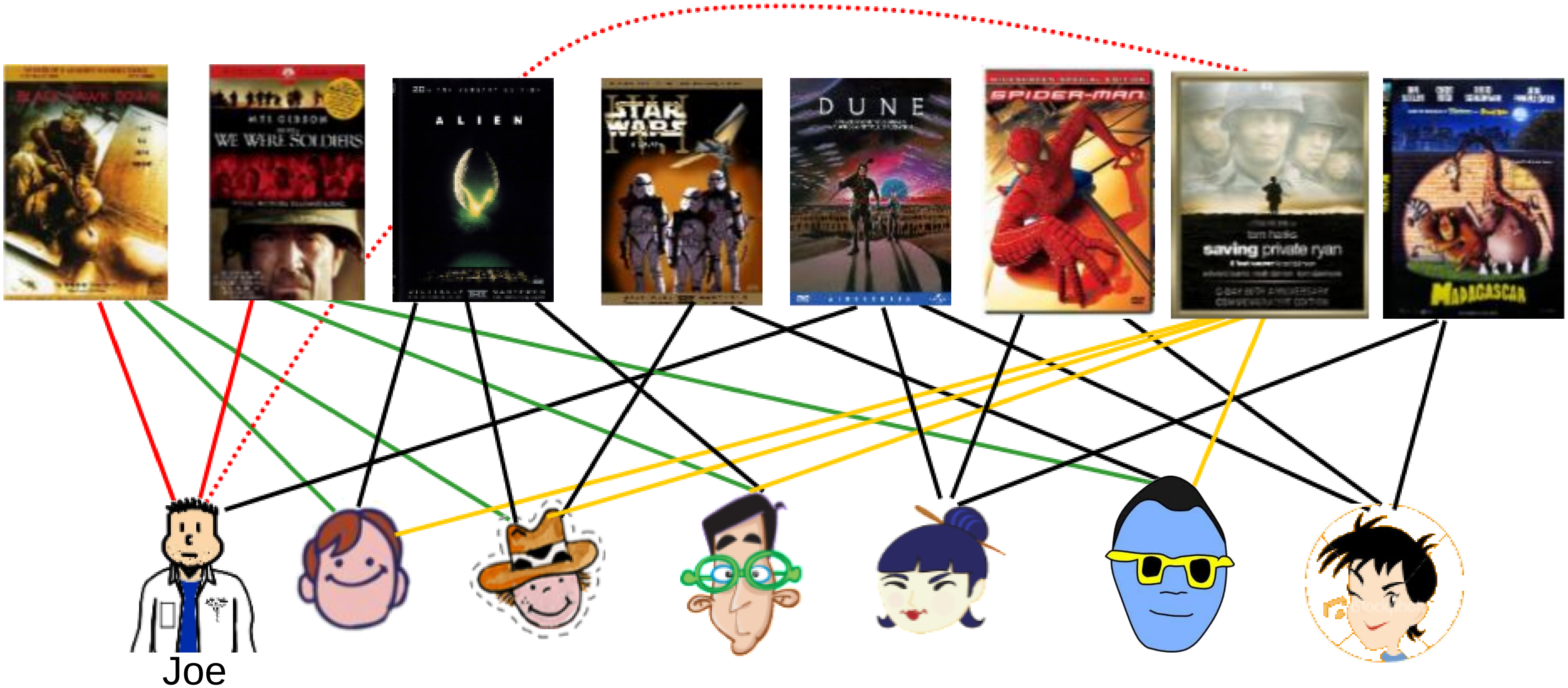


Y! Research

Recommender Systems

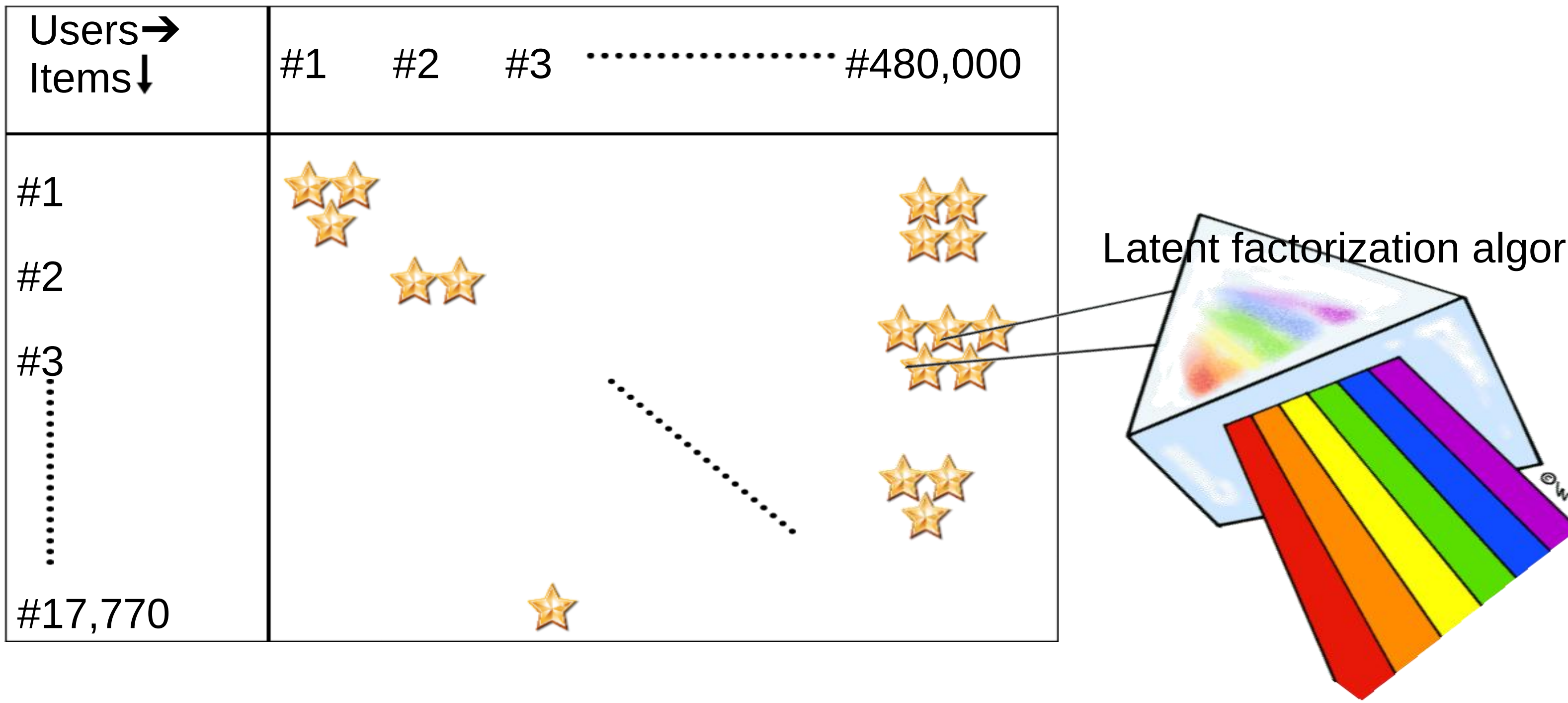
- Personalized recommendations of items (e.g., movies, TV, songs) to users
- Content based
  - Items scored on pre specified attributes
  - Users' interests estimated for same attributes
  - e.g., travel recommendations, eHarmony, Pandora
- Collaborative filtering (CF)
  - Does not require content information about items or user surveys
  - Infers relationships from purchases or ratings
  - Neighborhood based methods
  - Latent factor models

Neighborhood based collaborative filtering

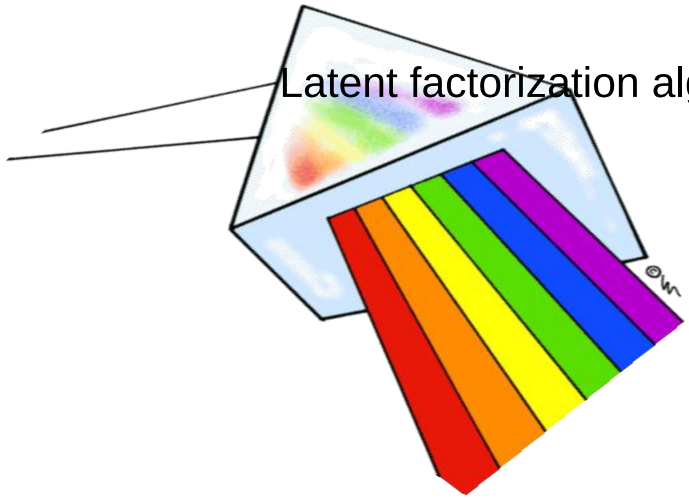


Latent factor methods

Users/items arranged in ratings space



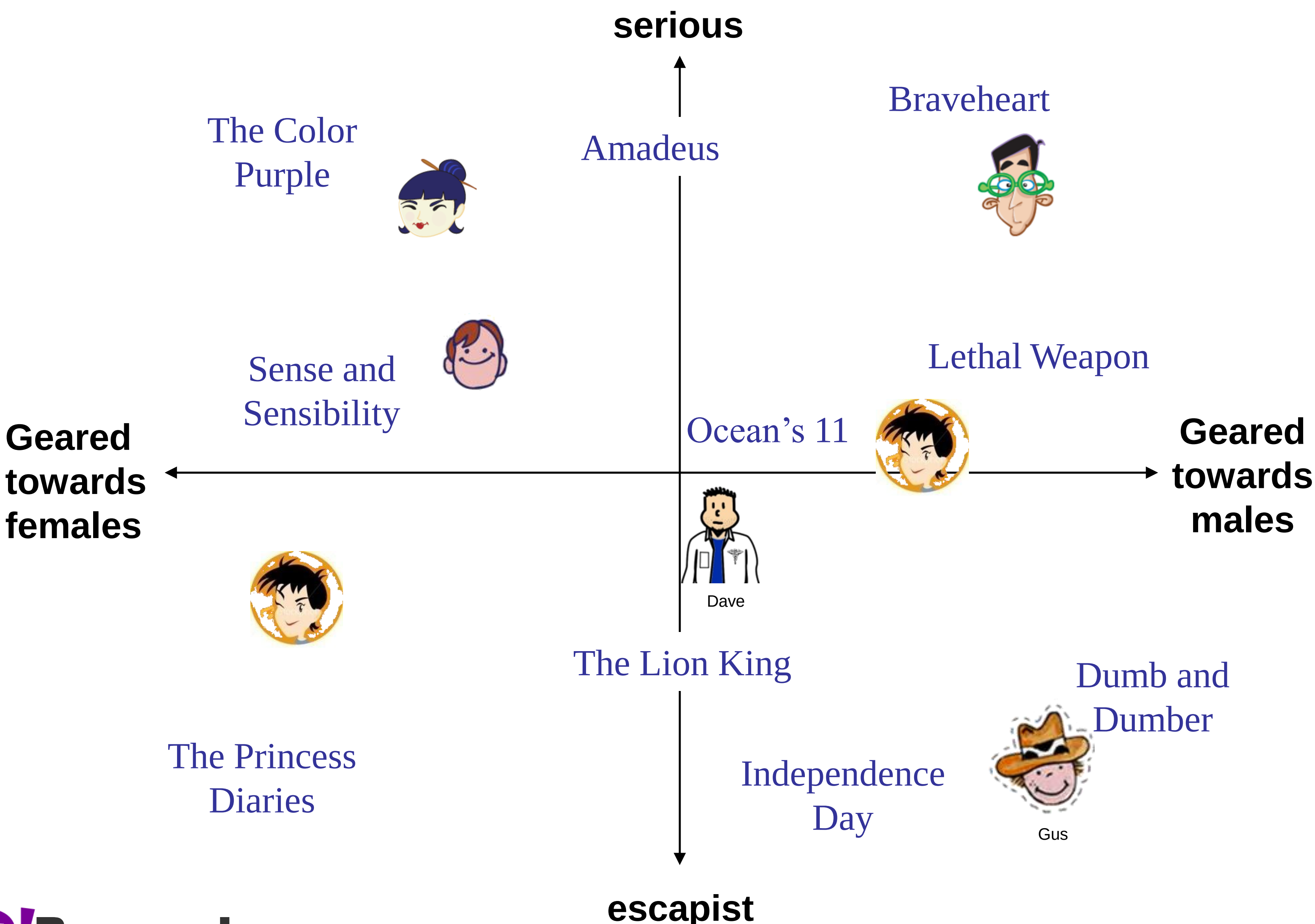
- Dim(Users)≠ Dim(Items) (E.g., 17,770-vs-480,000)
- Sparse data, with non-uniformly missing entries



Item-1	1.2	0.8	2.1	1.7	0.01	0.25
Item-2	1.1	0.01	1.19	1.35	1.25	0.37
Item-3	0.95	2.1	0.1	0.37	0.55	1.1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Item-17,770	0.44	0.12	0.43	0.76	0.87	0.17
User-1	0.08	0.49	0.37	1.2	0.67	1.3
User-2	0.77	1.1	0.04	0.97	1.05	1.95
User-3	0.19	0.13	0.88	1.2	1.87	1.1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
User-480K	1.4	1.9	1.4	0.37	0.95	0.7

Users/items arranged in joint dense latent factors space

A 2-D factor space



Y! Research

Basic matrix factorization model

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	

~

~

items

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

•

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

A rank-3 SVD approximation

Estimate unknown ratings as inner-products of factors:

users

items

1		3			5			5		4	
		5		?		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	

~

items

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

•

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

A rank-3 SVD approximation

Estimate unknown ratings as inner-products of factors:

users

items

1		3			5			5		4	
		5		?		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	

~

~ items

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

•

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

A rank-3 SVD approximation

Estimate unknown ratings as inner-products of factors:

users

items

1		3			5			5		4	
		5		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	

~

~ items

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

•

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

A rank-3 SVD approximation

Matrix factorization as a cost function

Rating prediction: $\hat{r}_{ui} = p_u^T q_i$

$$\mathrm{Min}_{p_*,q_*} \sum_{\text{known } r_{ui}} \Big(r_{ui} - \underbrace{p_u^T q_i}_{\text{prediction}}\Big)^2 + \underbrace{\lambda\left(\|p_u\|^2 + \|q_i\|^2\right)}_{\text{regularization}}$$

p_u - user-factor of u

q_i - item-factor of i

r_{ui} - rating by u for i

- Optimize by either stochastic gradient-descent or alternating least squares

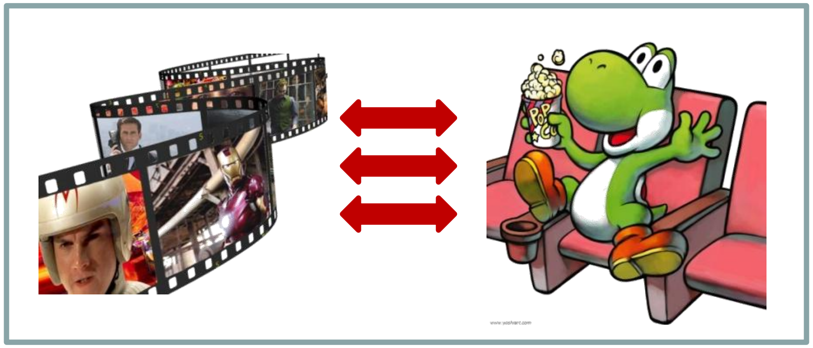
Y! Research

Talk outline

- Recommender systems – a quick intro
  - Neighborhood methods
  - Matrix factorization methods
- Biases and temporal dynamics
- Bootstrapping a recommender – ratings elicitation
- Y!Answers – combining multiple kinds of attributes and feedback
- Interpretation of user feedback: binary, numeric, or ordinal?
- Estimating confidence in recommendations
- KDD-Cup’2011

Components of a rating predictor

$$\hat{r}_{ui} = \underbrace{b_u}_{\text{user bias}} + \underbrace{b_i}_{\text{item bias}} + \underbrace{p_u^T q_i}_{\text{user-item interaction}}$$



Biases

- Separates users and movies
- Often overlooked
- Benefits from insights into users' behavior

User-item interaction

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

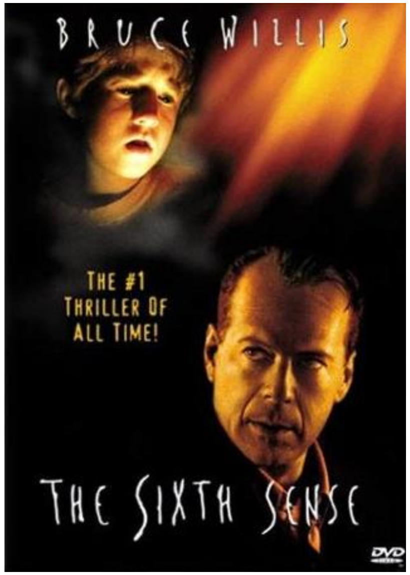
A bias estimator

- We have expectations on the rating by user u to item i, even without estimating u's attitude towards items like i

- Rating scale of user u
- Values of other ratings the user gave recently

- (Recent) popularity of item i
- Selection bias

Biases: an example

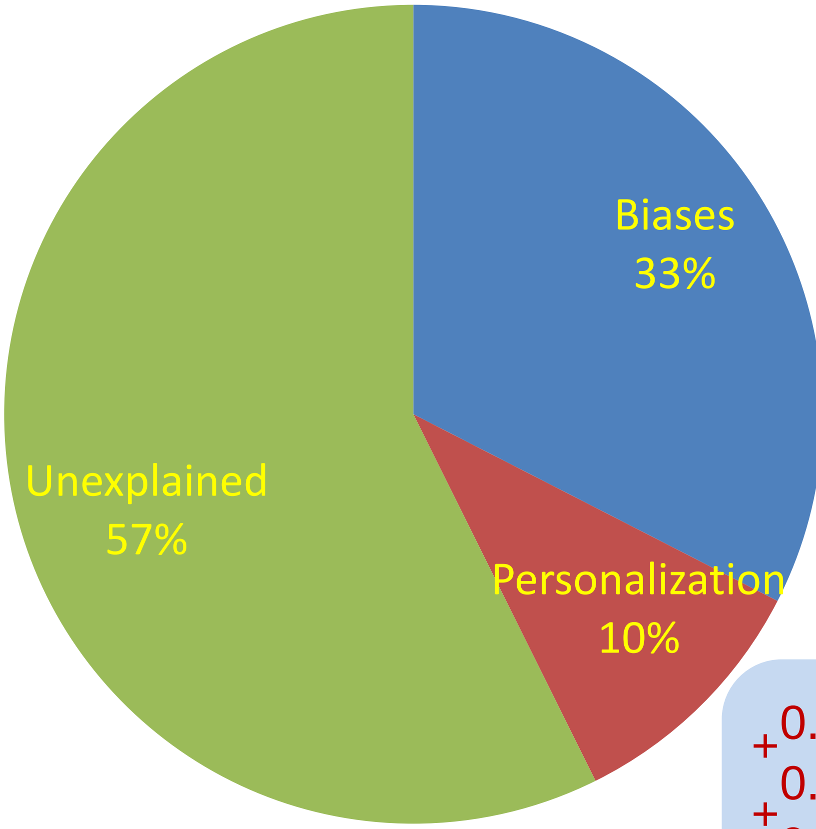


- Mean rating: 3.7 stars
- *The Sixth Sense* is 0.5 stars above avg
- *Joe* rates 0.2 stars below avg

➔Baseline estimation:
Joe will rate *The Sixth Sense* 4 stars

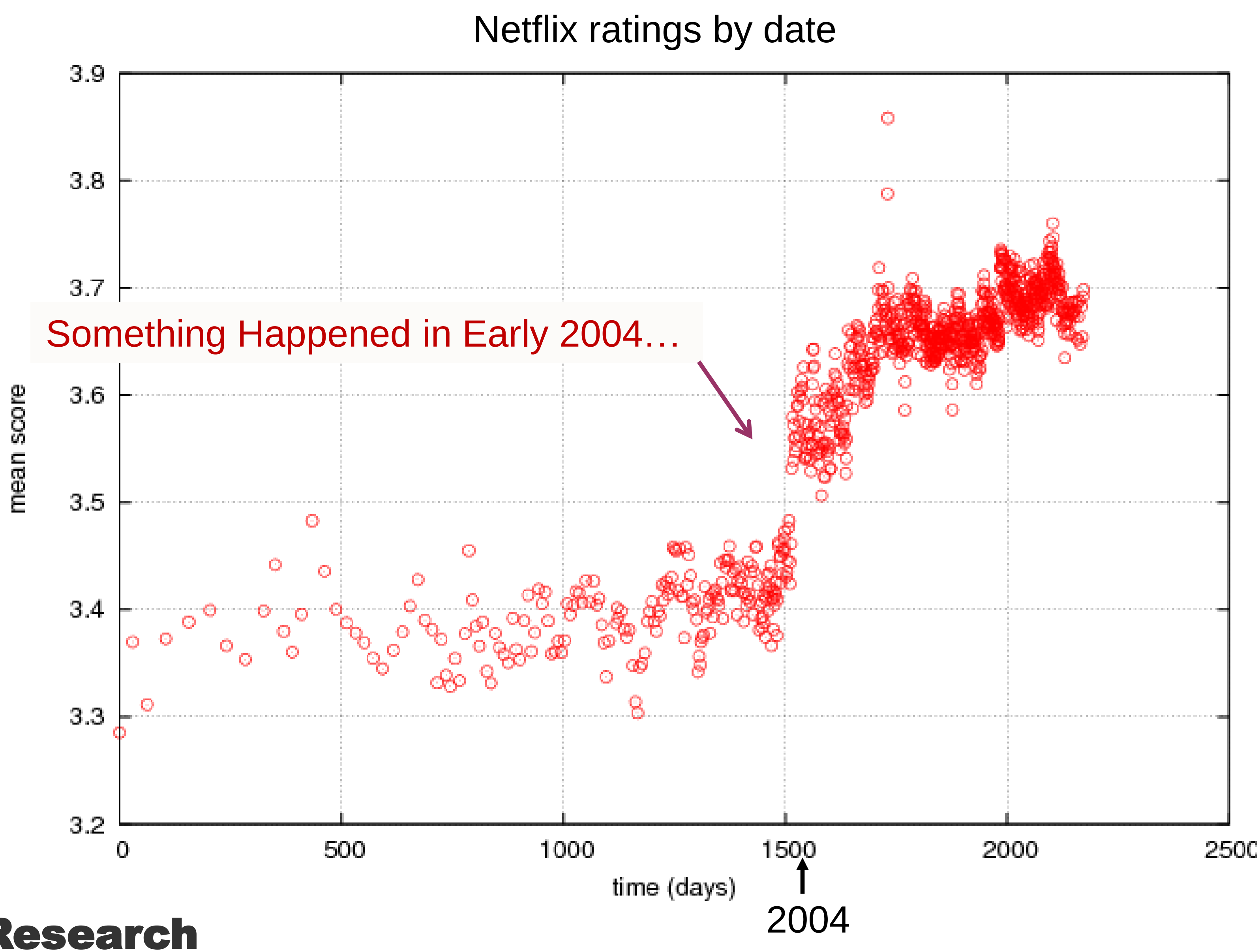
Biases matter!

Sources of Variance in Netflix data



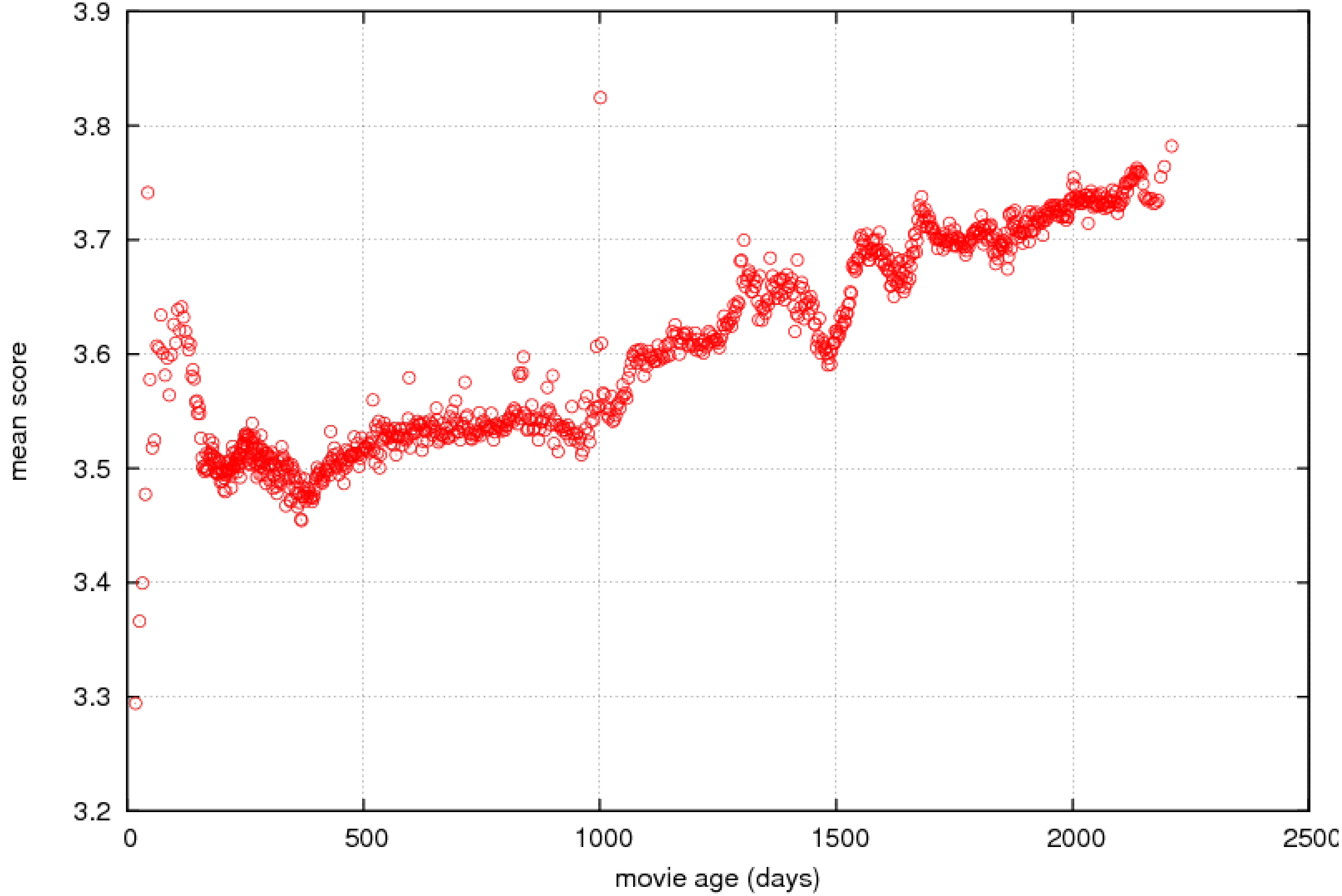
$$
\begin{array}{rl}
 & 0.732 \text{ (unexplained)} \\
+ & 0.415 \text{ (biases)} \\
+ & 0.129 \text{ (personalization)} \\
\hline
 & 1.276 \text{ (total variance)}
\end{array}
$$

Exploring Temporal Effects



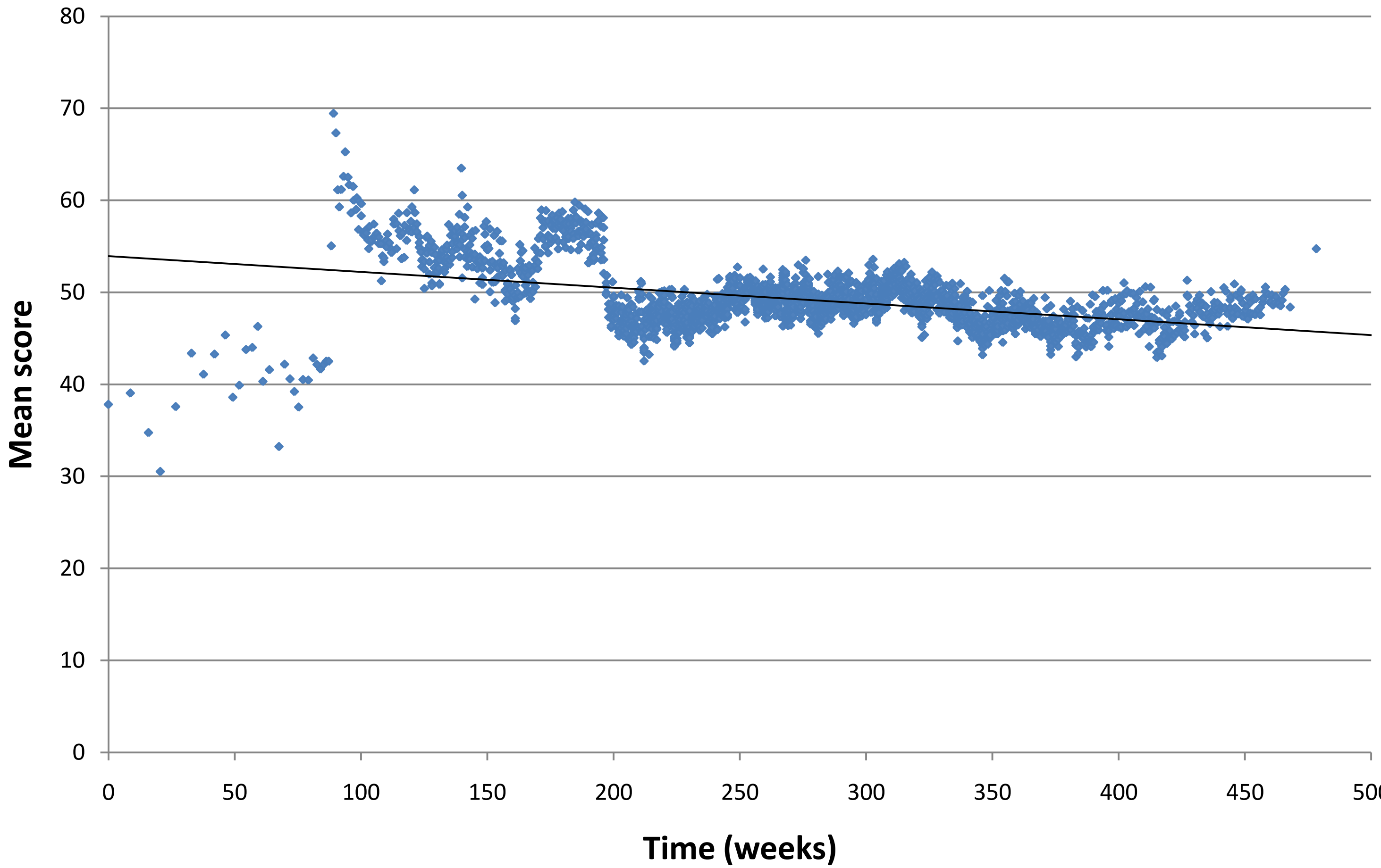
Y! Research

Are movies getting better with time?



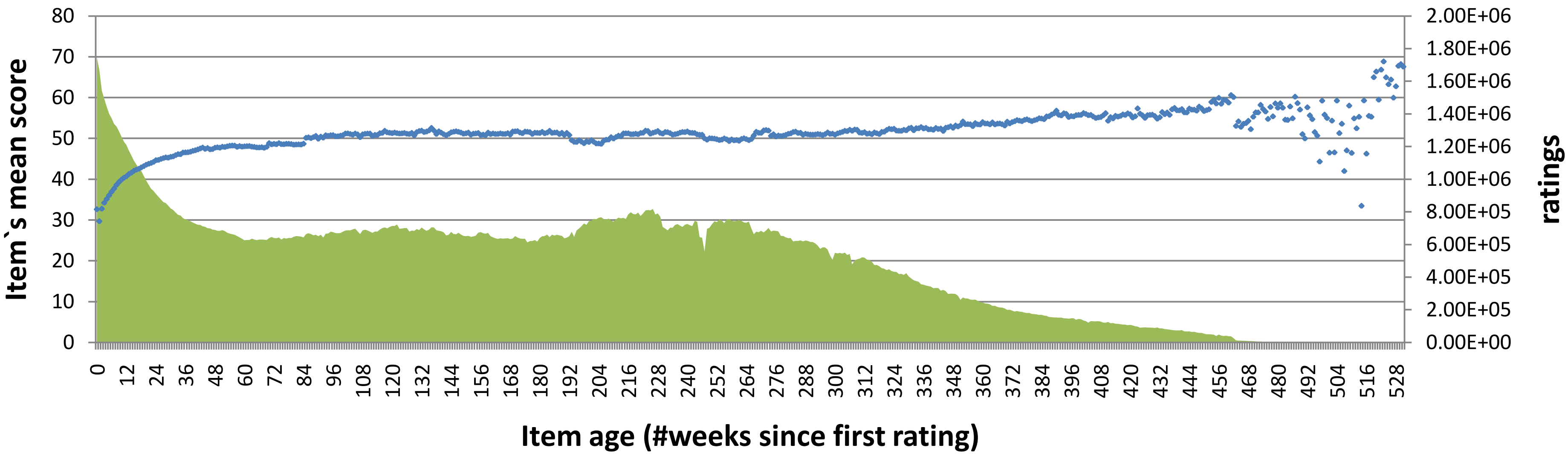
Y! Research

Yahoo! Music – ratings by time (total dataset)



Y!Research

Yahoo! Music items are also getting better with age…



Multiple sources of temporal dynamics

- Item-side effects:
  - Product perception and popularity are constantly changing
  - Seasonal patterns influence items' popularity
- User-side effects:
  - Customers redefine their taste
  - Transient, short-term bias; anchoring
  - Drifting rating scale
  - Change of rater within household

Introducing temporal dynamics into biases

- Biases tend to capture most pronounced aspects of temporal dynamic
- We observe changes in:
    1. Rating scale of individual users (user bias)
    2. Popularity of individual items (item bias)

$$\hat{r}_{ui} = b_u + b_i + p_u^T q_i$$

Add temporal dynamics

$$\hat{r}_{ui}(t) = b_u(t) + b_i(t) + q_i^T p_u$$

Y! Research

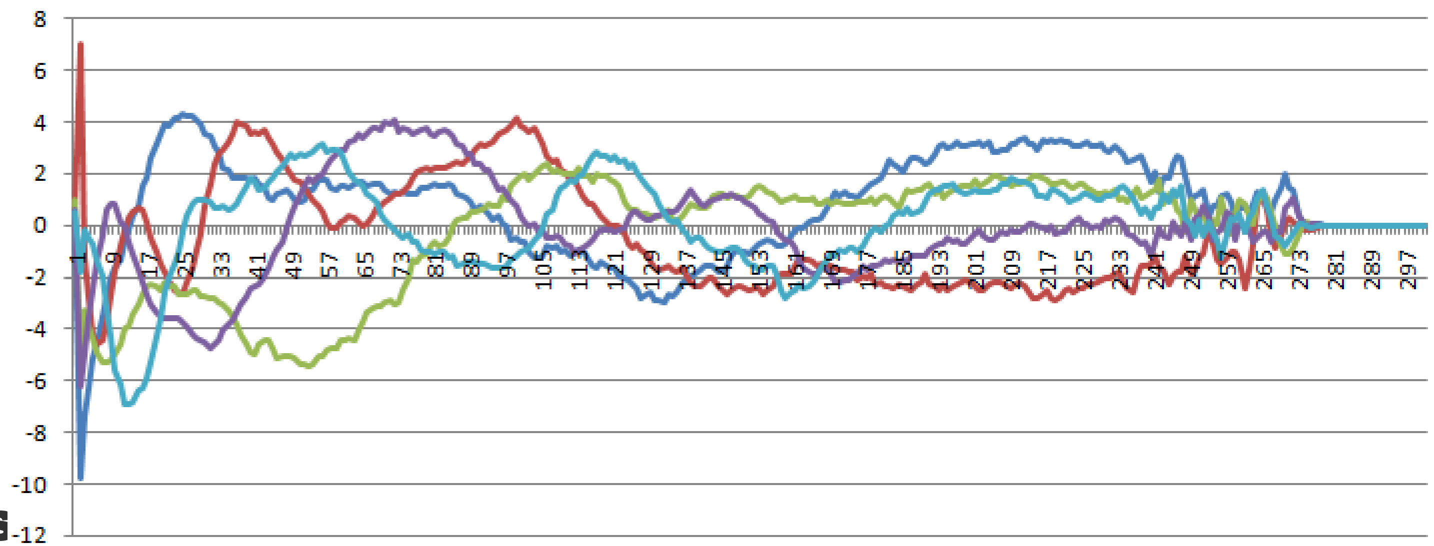
Item biases – modeling smooth long-term temporal change

- A time dependent item bias – function of weeks since first rating

$$b_i(t) \Rightarrow b_i + b_i\left(week(t)\right)$$

- Can be seen as a regression of *N* base functions *c* with item-dependent weights *x*

$$b_i\left(week\right) = \sum_{k=1}^{N} x_i(k) \cdot c(week, k)$$



User biases – modeling short-term, session-based temporal changes

- Short term effects are very influential on user ratings
  - Changes of mood
  - Short-term needs
  - Changes in identity
  - Anchoring effects
  - Drifting effects
- Longer term effects are less pronounced and harder to capture
- User session bias:

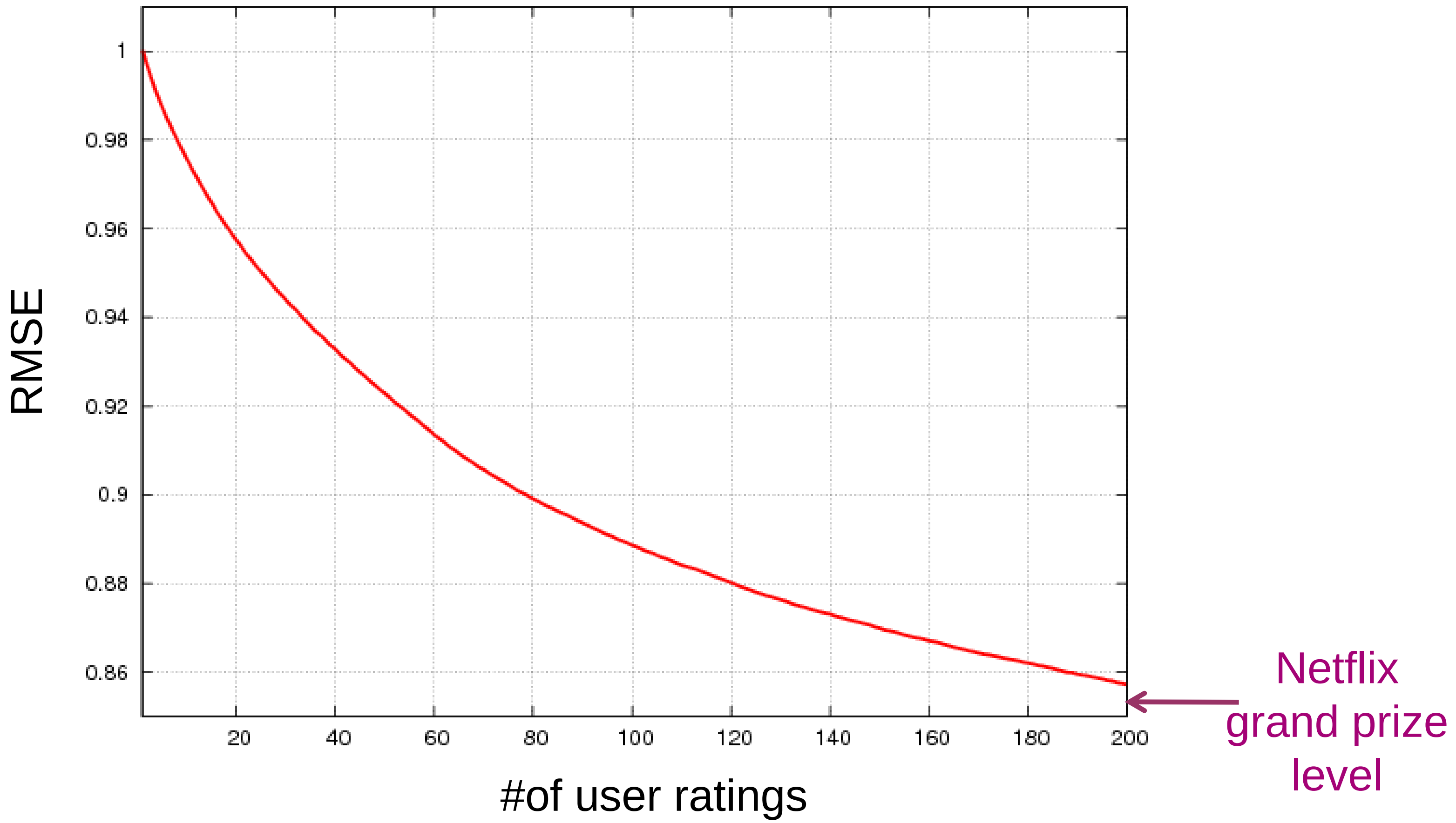
$$b_u(t) = b_u + b_{u,session(t)}$$

- User session bias cannot predict the future, but rather eliminate noise while training the model

Talk outline

- Recommender systems – a quick intro
  - Neighborhood methods
  - Matrix factorization methods
- Biases and temporal dynamics
- Bootstrapping a recommender – ratings elicitation
- Y!Answers – combining multiple kinds of attributes and feedback
- Interpretation of user feedback: binary, numeric, or ordinal?
- Estimating confidence in recommendations
- KDD-Cup'2011

More user input > better algorithms



Y! Research

Eliciting user feedback

- Observation:
  New users are hard to predict, yet are most judgmental
- Challenge:
  Quickly accumulate feedback that allows profiling the user
- Solution:
  Ratings elicitation through an interview

Questions

Show [Four Movies] per page

Tell us your opinion on these movies...

Lord of the Rings: The Return of the King: Extended Edition (2003)

- ( ) Like
- ( ) Don't like
- (•) Don't have an opinion

Finding Nemo (Full-screen) (2003)

- ( ) Like
- ( ) Don't like
- (•) Don't have an opinion

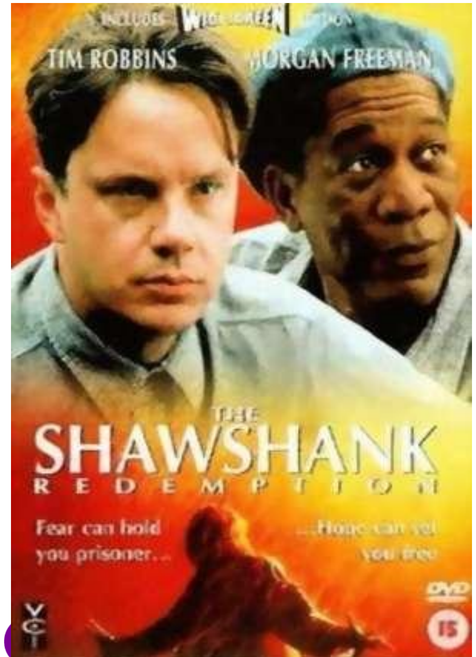
Something the Lord Made (2004)

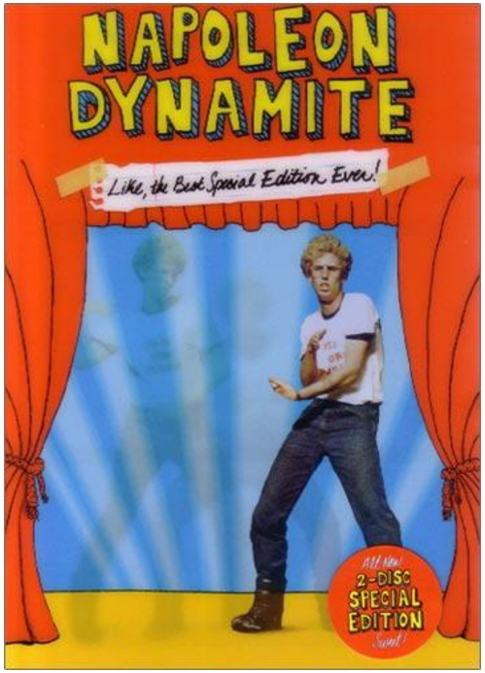
- ( ) Like
- ( ) Don't like
- (•) Don't have an opinion

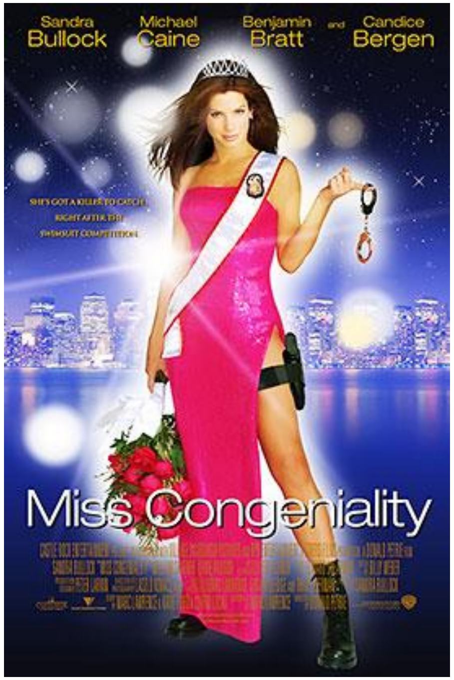
Which items to ask about?

General guidelines:

- Popularity – items familiar to most users
- Contention – controversial items reflect distinct tastes
- Correlation – only interested in items indicative on others



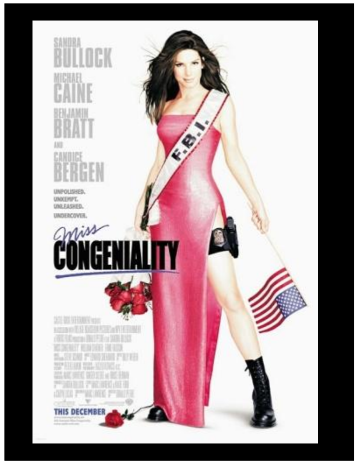


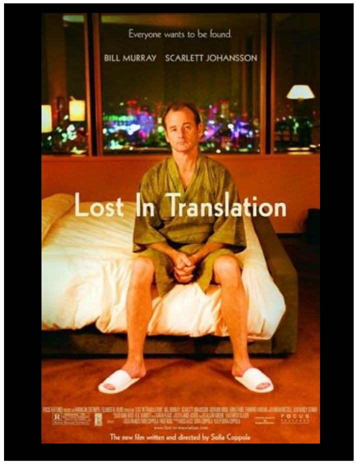


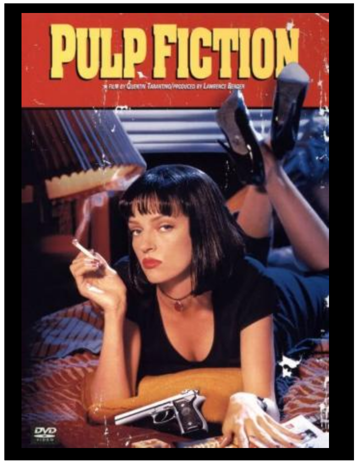
Picking items to ask about – a principled solution

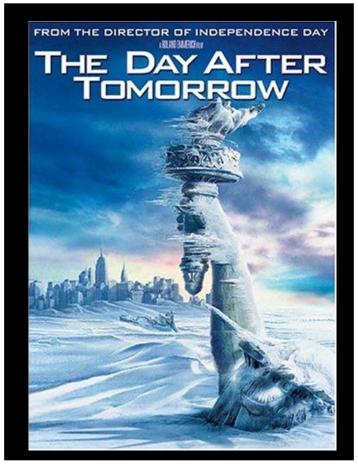
- Popularity, contention and correlation may conflict with each other – how to balance them?
- Other considerations also influence user experience
  - E.g. quality of the full set of items (don't ask on two similar items twice…)
- Our solution [CIKM'10]:
  Pick size-k item set S optimizing subsequent performance of algorithm *A*:
  $$S = argmin_{S \subset Items, |S|=k} RMSE\left(\mathcal{A}(S)\right)$$
- Doesn't require balancing multi-objectives
- A greedy optimization procedure

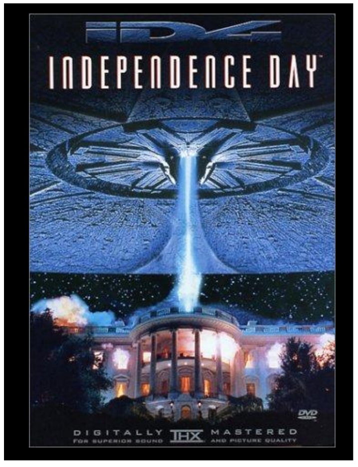
Most Discriminating Movies









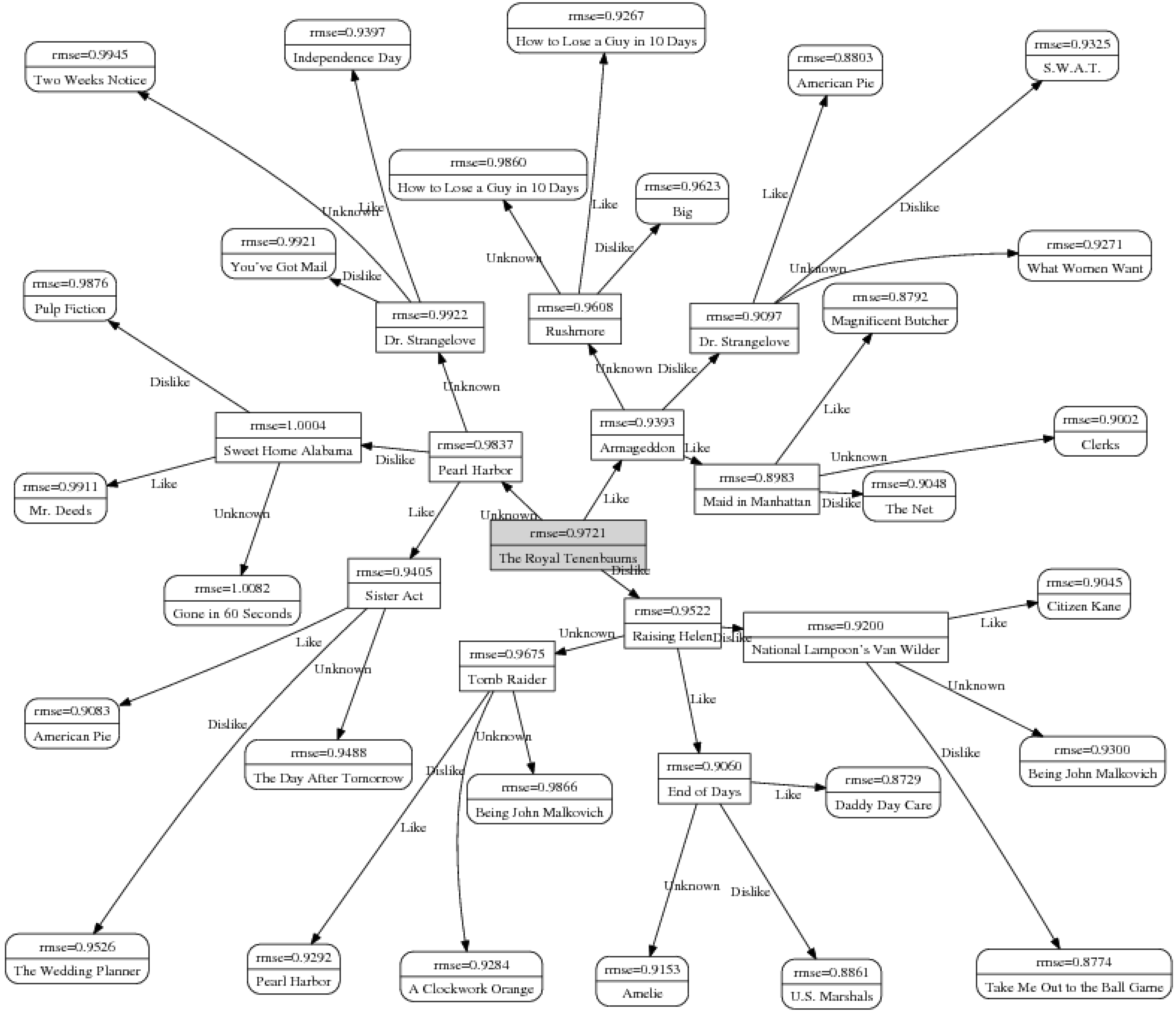


A better alternative: adaptive rating elicitation

- Adapt the interview process to the input already given by the user
- Each user is asked on different items
- Users can be profiled through far less questions
- A decision tree allows representing all interview paths

Decision tree for rating elicitation

- Construct a RMSE-minimizing ternary decision tree [WSDM'11]
- Each node corresponds to a group of users
- All users at a node receive same recommendations
  ➔ Error measure for node's quality (RMSE)
- Each internal node is split by its pivot item

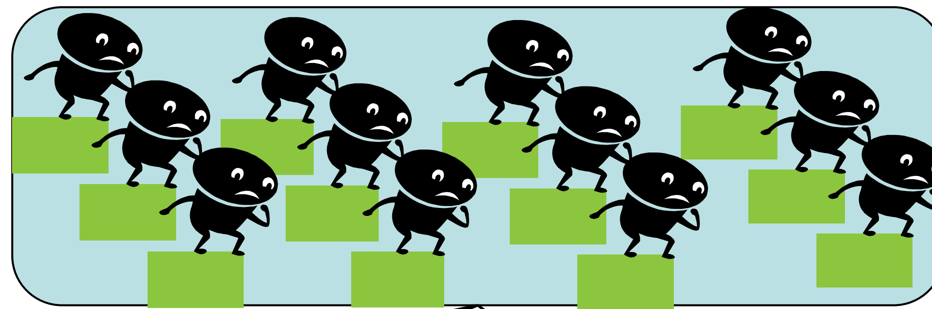


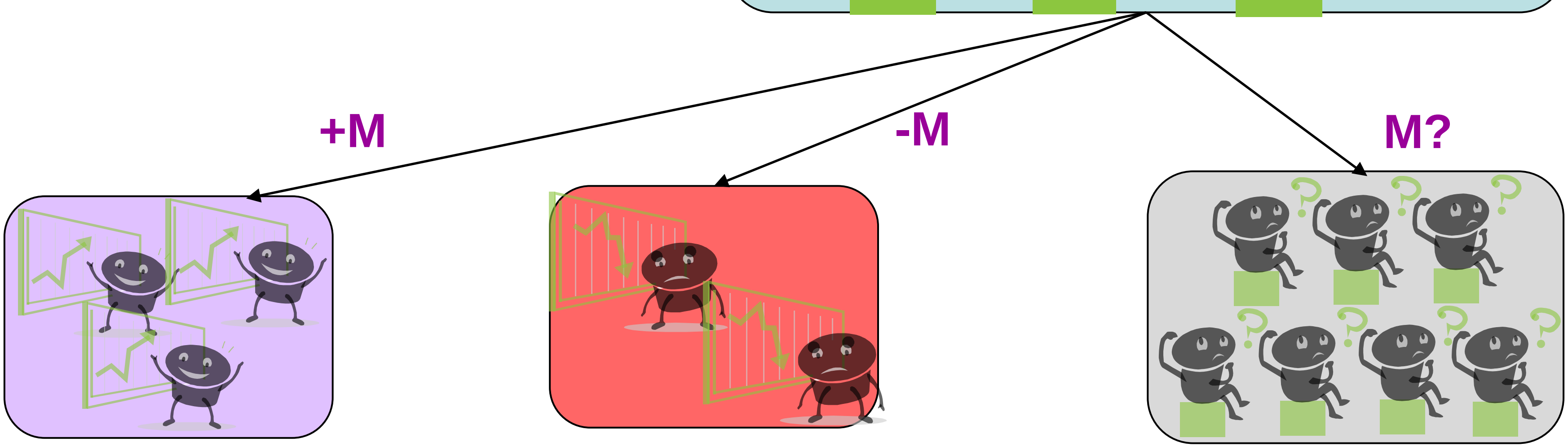
rmse=0.9721
The Royal Tenenbaums
rmse=0.9837
Pearl Harbor
rmse=0.9393
Armageddon
rmse=0.9522
Raising Helen
rmse=0.9922
Dr. Strangelove
rmse=1.0004
Sweet Home Alabama
rmse=0.9405
Sister Act
rmse=0.9608
Rushmore
rmse=0.9097
Dr. Strangelove
rmse=0.8983
Maid in Manhattan
rmse=0.9675
Tomb Raider
rmse=0.9200
National Lampoon's Van Wilder
rmse=0.9060
End of Days
rmse=0.9945
Two Weeks Notice
rmse=0.9397
Independence Day
rmse=0.9921
You've Got Mail
rmse=0.9876
Pulp Fiction
rmse=0.9911
Mr. Deeds
rmse=1.0082
Gone in 60 Seconds
rmse=0.9860
How to Lose a Guy in 10 Days
rmse=0.9267
How to Lose a Guy in 10 Days
rmse=0.9623
Big
rmse=0.8803
American Pie
rmse=0.9325
S.W.A.T.
rmse=0.9271
What Women Want
rmse=0.8792
Magnificent Butcher
rmse=0.9002
Clerks
rmse=0.9048
The Net
rmse=0.9045
Citizen Kane
rmse=0.9300
Being John Malkovich
rmse=0.8774
Take Me Out to the Ball Game
rmse=0.8729
Daddy Day Care
rmse=0.8861
U.S. Marshals
rmse=0.9153
Amelie
rmse=0.9284
A Clockwork Orange
rmse=0.9866
Being John Malkovich
rmse=0.9292
Pearl Harbor
rmse=0.9526
The Wedding Planner
rmse=0.9488
The Day After Tomorrow
rmse=0.9083
American Pie
Like
Dislike
Unknown

Efficiently building a decision tree

- Major task: evaluate movie **M** as pivot for current node

Large population with known RMSE (*), asked about a certain movie M

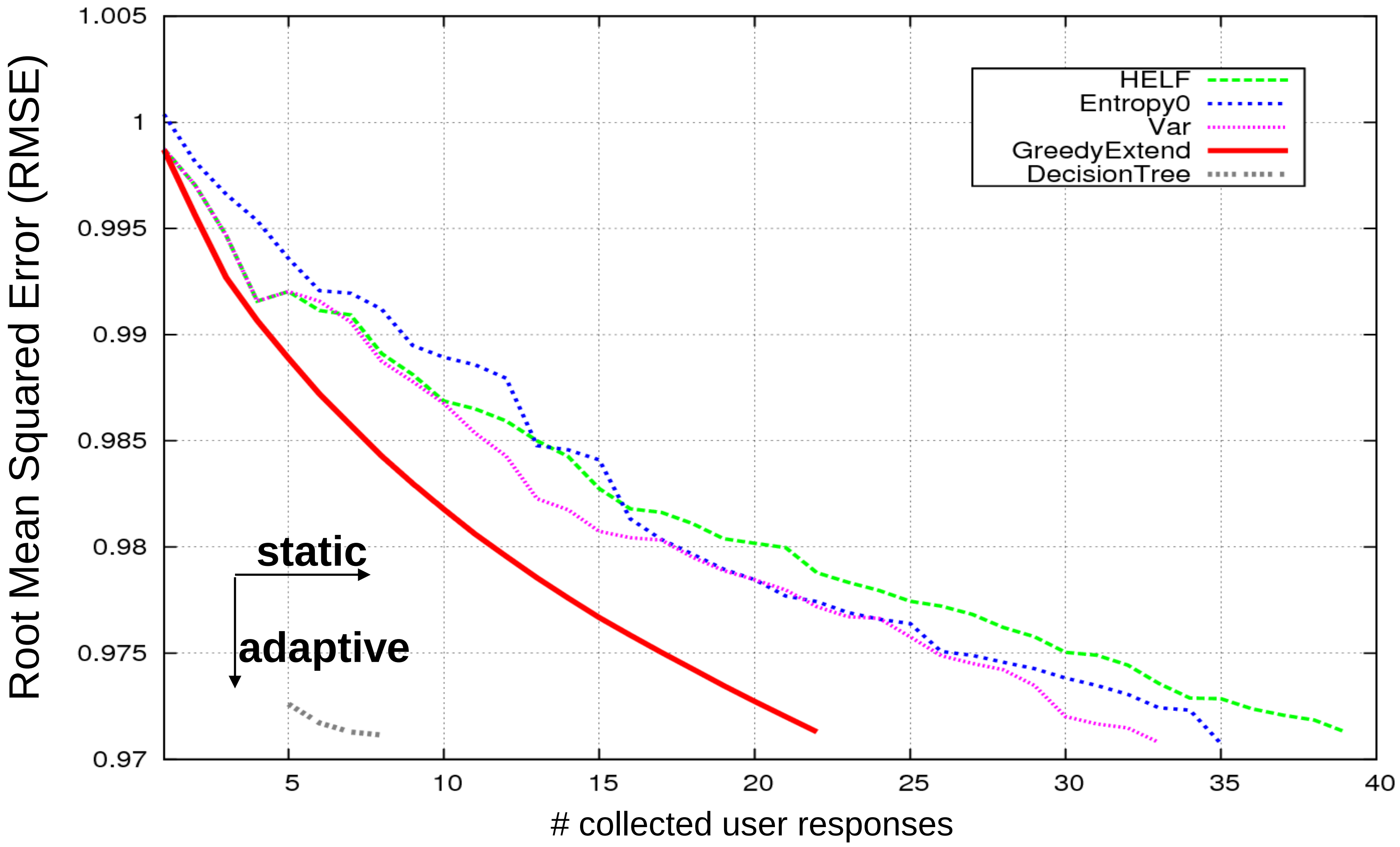




Small populations: compute RMSE (+,-) explicitly

Large population: infer RMSE from RMSE (*,+,-)

Adaptive questionnaire significantly outperforms a static one



Y! Research

Talk outline

- Recommender systems – a quick intro
  - Neighborhood methods
  - Matrix factorization methods
- Biases and temporal dynamics
- Bootstrapping a recommender – ratings elicitation
- Y!Answers – combining multiple kinds of attributes and feedback
- Interpretation of user feedback: binary, numeric, or ordinal?
- Estimating confidence in recommendations
- KDD-Cup'2011

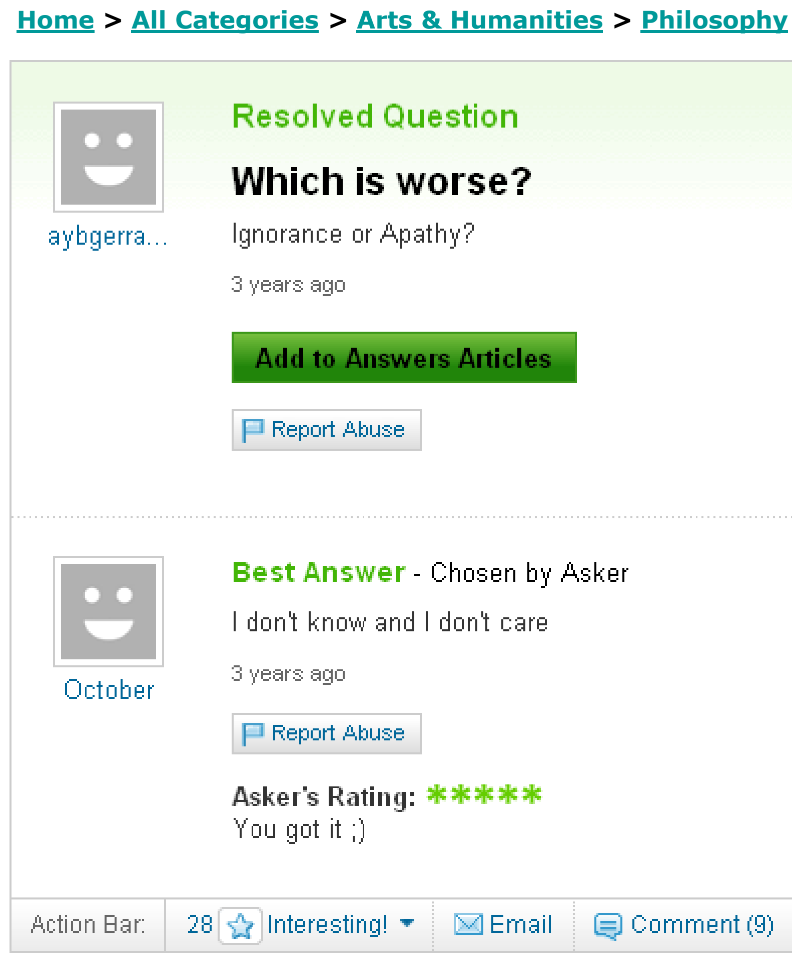
A multi-channel recommendation system

Multiple channels

- Data scarcity is a constant problem for recommenders, especially when dealing with new users and new/rare items
- We would like to leverage all kinds of signals available to us
- Many types of item attributes:
  - Item category, name, tags, editorial description, user generated content
  - Each of a different quality and quantity
- Many types of (implicit) user feedback on items
  - Views, clicks, votes, saves, purchase, writing,…
  - Each of a different quality and quantity
- How to combine all different signals together, while accounting for their different significance levels?
  - We don’t want to arbitrarily weight signals / channels
  - Also don’t want to discard valuable signal
- A test case: Yahoo! Answers recommender system

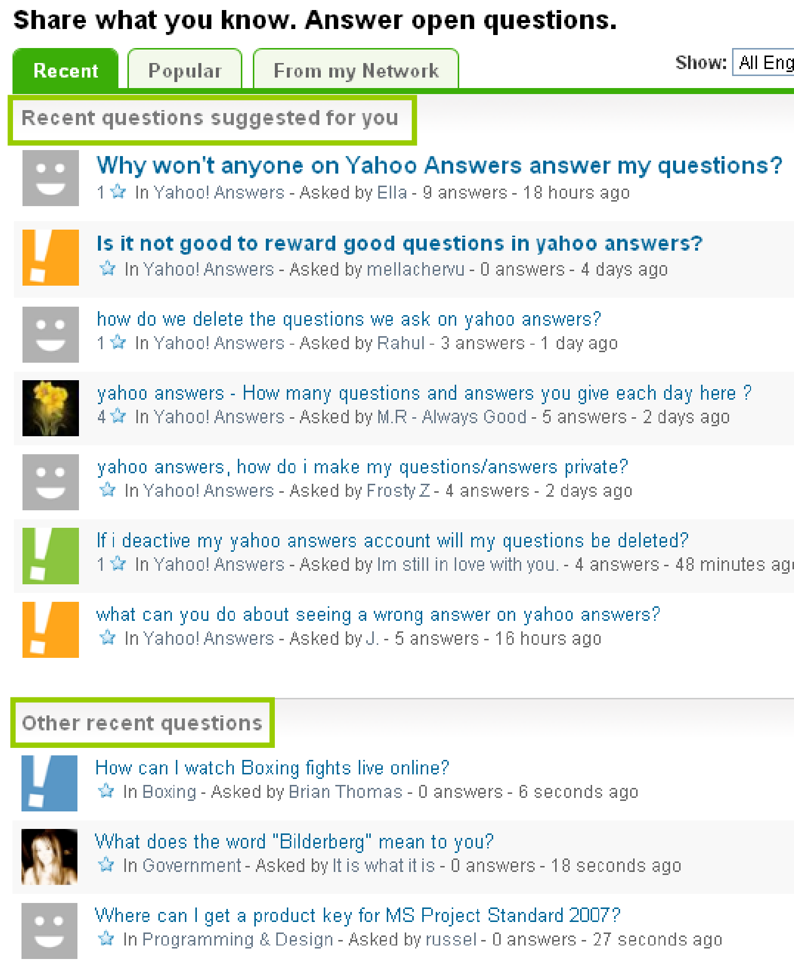
Yahoo! Answers

- Community Question Answering
  - users ask questions, other users answer
- The most popular site:
  - 2 million active users per month
  - 200 million viewers per month
  - 2-3 new questions per second
  - 3 answers per question on average
    - More than 1 billion answers
- Question lifecycle:
  - Opened (in a category): users can answer
  - In voting: voting on best answer
  - Resolved: Best answer was decided
- User activities:
  - *ask, answer, vote on best answer, vote on question, comment, report abuse, etc.*
- Problem: many questions are not answered well or unanswered



Question Routing

- Goal: Increase the chances of questions being answered to the asker's satisfaction in Y! Answers
- Solution: match between questions and users that can answer them well
  - given a new question, push it to the "best" potential answerers
  - given a user, present the "best" questions for her/him to answer
- Technology: recommender system
  - build profiles for questions and users
  - Measure match between a user and a question
  - Emphasis on multi-channel integration



Question Profile

- Extract values from various question fields
  - Textual: words
  - Category: question category
  - Users (social): users that interacted with the question

- [title : *worse*]
  [body : *ignorance*, *apathy*]
  [answer : *know*, *care*]
  [category : *Philosophy*]
  [asker : *u83*]
  [answerer : *u7*]

- Key point: keep fields separated
  - Don't mix words from title with words from body

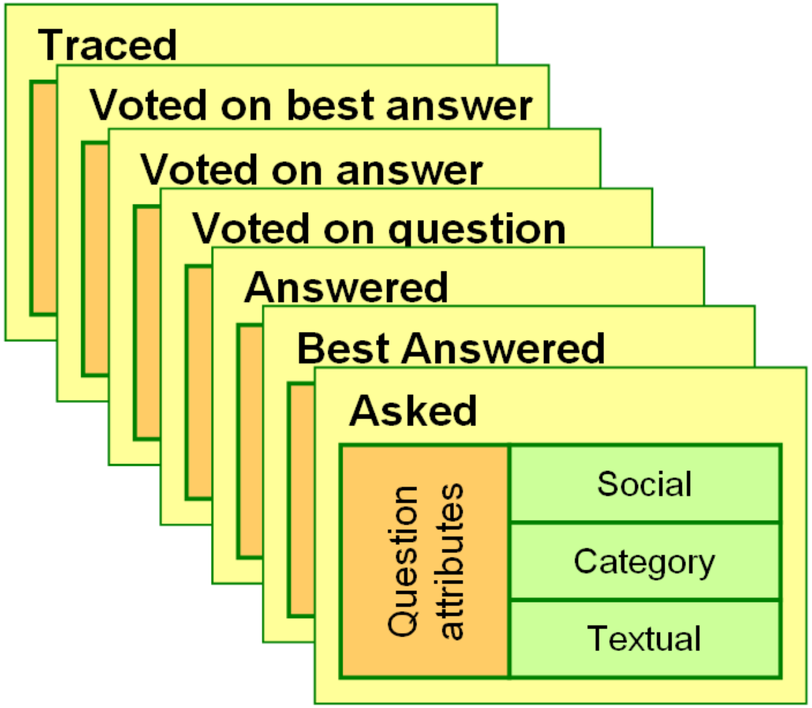
User IDs (social interactions) attributes	Question tracers
	Best answer selectors
	Answer voters
	Question voters
	Answerers
	Best answerer
	Asker
Categ. attributes	Grandparent category
	Parent category
	Exact category
Textual attributes	Other answers
	Best answer
	Body
	Title

User Profile

- Aggregate all values of questions the user interacted with
  - Questions asked, questions answered, questions voted on

- [asked : title : *worse*]
  [asked : answer : *know*, *care*]
  [voted : title : *basketball*, *baseball*]
  [answered : answer : *football*]

- Key point: keep interactions separated
  - Don't mix titles of asked questions with titles of answered questions



User-Question Pair Model

- Reminder: assess match of a potential answerer to a question
- Generate features for a *user-question* pair
  - Sum shared values between user and question attributes

- Q: **What is the difference between basketball, baseball, volleyball, soccer, and football?**
  - <U:voted:title , Q:title> = 2 (*basketball*, *baseball*)
  - <U:answered:answer , Q:title> = 1 (*football*)

- Over 500 combined features derived when interacting user- and question-profile
- Train a classifier – learning the match measure
  - Significance of each channel is learned from the data

Talk outline

- Recommender systems – a quick intro
  - Neighborhood methods
  - Matrix factorization methods
- Biases and temporal dynamics
- Bootstrapping a recommender – ratings elicitation
- Y!Answers – combining multiple kinds of attributes and feedback
- Interpretation of user feedback: binary, numeric, or ordinal?
- Estimating confidence in recommendations
- KDD-Cup’2011

Interpretation of ratings

- Most methods take *explicit feedback* as numerical
  - E.g., a star-scale is taken as numbers between 1 and 5.
- *Implicit feedback* is usually taken as binary values
  - E.g., “bought”-vs-“didn’t buy”, “viewed”-vs-”didn’t view”
- These views align well with computational convenience, but might be too restrictive:
  - Different levels of user actions:
    “view” < “click” < “add to wish list” < “add to cart” < “purchase”
  - Some systems collect ratings as letters: “A+”, “A”, “A-”, “B+”, “B”,…,”F”

Jones

YAHOO! MOVIES

Search

Earth (2009)

Like 98 retweet 1

- Movie Main Page

Movie Overview

- Movie Details
- Showtimes & Tickets
- DVD/Video Info
- Trailers & Clips
- Cast and Credits
- Awards & Nominations

Reviews and Previews

- Critics Reviews
- User Reviews

Photos

- Premiere Photos
- Movie Stills

Community

- Message Board

Shopping

- Buy the DVD/Video

Other Resources

- Web Sites

What's New

Exclusive - Watch a clip from Disneynature's 'Earth'

earth

Watch the Trailer

GET IT from BLOCKBUSTER

The Critics:	My Grade:
B	Rate this Movie!
10 reviews	A+ Submit
Yahoo! Users:	write a review
B+	
2421 ratings	

The story of three animal families and their amazing journeys across the planet we all call home. The film combines rare action, unimaginable scale and impossible locations by capturing the most intimate moments of our planet□s wildest and most elusive creatures.

Genres: Action/Adventure, Art/Foreign and Documentary
Running Time: 1 hr. 30 min.
Release Date: April 22nd, 2009
MPAA Rating: G
U.S. Box Office: $32,001,863

See Full Details

Cast and Credits

Starring: Patrick Stewart, James Earl Jones
Directed by: Alastair Fothergill
Produced by: Andre Sikojev, Nikolaus Weil, Stefan Beiten

See Full Cast and Credits

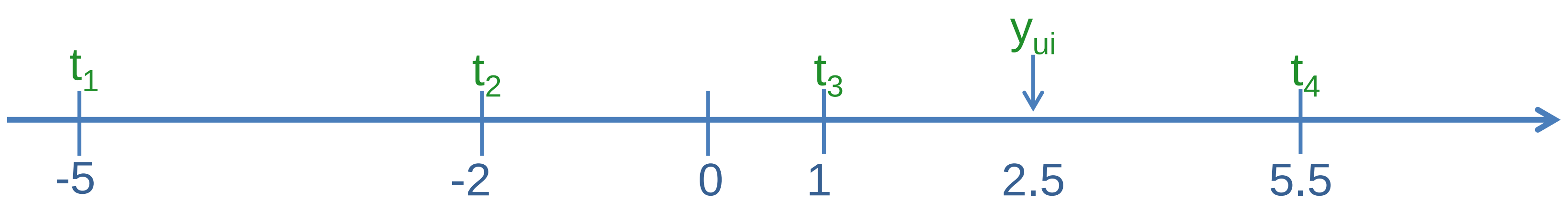
An ordinal rating scale

An ordinal rating scale

- Ratings are taken as ordinal (ordered values)
  - E.g.: "1 star" < "2 stars" < "3 stars"< "4 stars" < "5 stars", but no notion of numerical distance among ratings
- A general relaxation of the more restrictive numerical/binary views; can capture user feedback such as:
  - "view", "click", "add to wish list", "add to cart", "purchase"
  - "A+", "A", "A-", "B+", "B",...,"F"
- Better fits users' intention
  - Even when ratings are "pseudo-numeric", e.g. stars, users view them as ordinal
  - Different users employ different internal rating scales:
    For many: "3 stars" rating may be closer to "1-2 stars " than to "4 stars"

An ordinal ranking model

- Model has two kinds of parameters: thresholds (t_1, t_2,…) and internal scores (y_{ui}); both are automatically learned from the data
- For example, given a user u and item i, at a 5-star rating scale
  - Set user-specific thresholds
  - Set internal score
  - Predict rating (r_{ui}) by computing implied probabilities



$$p(r_{ui} \leq k) = 1/(1+\exp(y_{ui} - t_k))$$

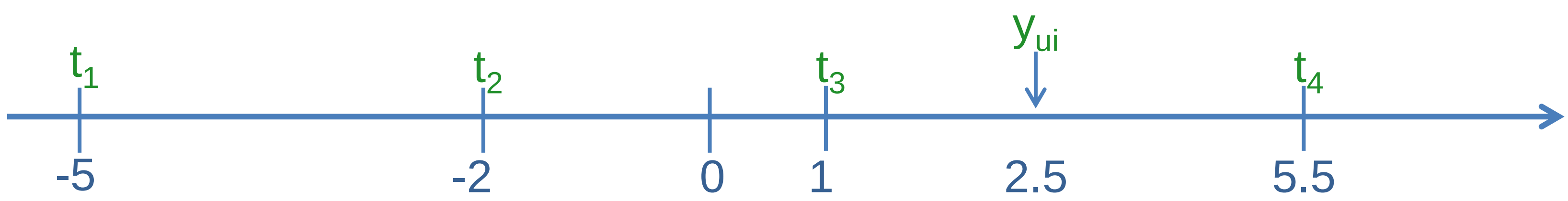
$$p(r_{ui} = k) = p(r_{ui} \leq t_k) - p(r_{ui} \leq t_{k-1})$$

$$p(r_{ui} \leq t_0) = 0, \quad p(r_{ui} \leq t_5) = 1$$

Y! Research

An ordinal ranking model

- Model has two kinds of parameters: thresholds (t_1, t_2,…) and internal scores (y_{ui}); both are automatically learned from the data.



$$p(r_{ui} \le k) = 1/(1+\exp(y_{ui} - t_k))$$

$$p(r_{ui} = k) = p(r_{ui} \le t_k) - p(r_{ui} \le t_{k-1})$$

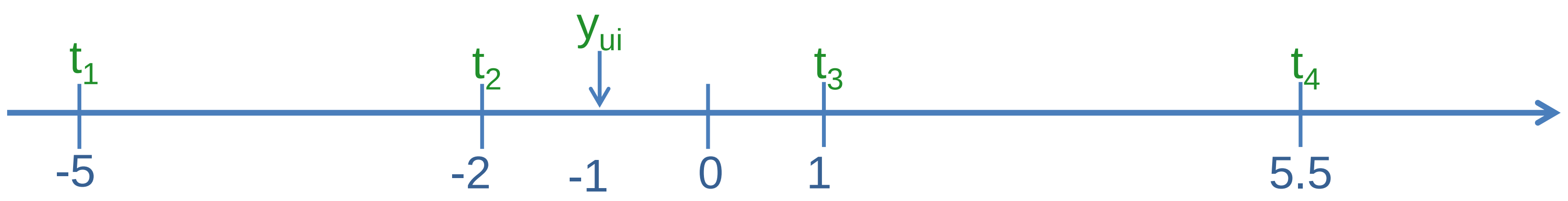
$$p(r_{ui} \le t_0) = 0, \;\; p(r_{ui} \le t_5) = 1$$

	k=1	k=2	k=3	k=4	k=5
$p(r_{ui} \le k)$	0.000553	0.010987	0.182426	0.952574	1
$p(r_{ui} = k)$	0.000553	0.010434	0.171439	0.770149	0.047426

Outputs a full distribution of scores

An ordinal ranking model

- Model has two kinds of parameters: thresholds (t_1, t_2,…) and internal scores (y_{ui}); both are automatically learned from the data.



$$p(r_{ui} \le k) = 1/(1+\exp(y_{ui} - t_k))$$

$$p(r_{ui} = k) = p(r_{ui} \le t_k) - p(r_{ui} \le t_{k-1})$$

$$p(r_{ui} \le t_0) = 0, \quad p(r_{ui} \le t_5) = 1$$

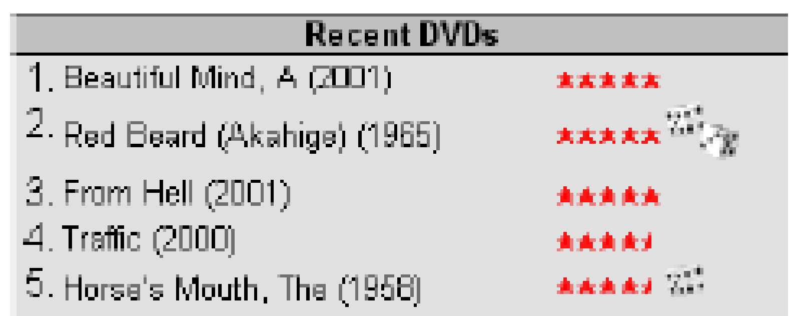
	k=1	k=2	k=3	k=4	k=5
$p(r_{ui} \le k)$	0.017986	0.268941	0.880797	0.998499	1
$p(r_{ui} = k)$	0.017986	0.250955	0.611856	0.117702	0.001501

Talk outline

- Recommender systems – a quick intro
  - Neighborhood methods
  - Matrix factorization methods
- Biases and temporal dynamics
- Bootstrapping a recommender – ratings elicitation
- Y!Answers – combining multiple kinds of attributes and feedback
- Interpretation of user feedback: binary, numeric, or ordinal?
- Estimating confidence in recommendations
- KDD-Cup'2011

Estimating confidence in recommendations

- Many considerations involve which product(s) to suggest
  - Predicted rating, diversity, novelty, profitability, context,…
- Here, we discuss confidence
- Even when predicted ratings are equal, the system may have different certainty level in each of them
- Bears impact on end-user experience
- Allow system designers to flexibly pick more confident items (e.g., minimize risk of disappointment)
- Enables confidence-aware combinations of multiple methods



Confidence Displays; McNee et al, GroupLens

Computing confidence

- Deviation from the total population mean score
  - Differentiate conventional recommendation from novel ones
- Baseline confidence estimators – capture hard users/items:
  - #of ratings for item / user
  - Standard deviation of item /user ratings
  - Disadvantages:
    - Not-personalized (would just stay away from controversial items, etc.)
    - Disregards the prediction algorithm

Probability-based confidence

- Employ recommenders outputting full probability distribution of ratings
  - For example, the described ordinal rating recommender, or RBM

	k=1	k=2	k=3	k=4	k=5
$p(r_{ui}=k)$	0.000553	0.010434	0.171439	0.770149	0.047426

Probability-based confidence

- Employ recommenders outputting full probability distribution of ratings
- Associate confidence with probability concentration
  - Use entropy, standard deviation, or Gini impurity of prob. distribution

An example

- Prediction of item *A*: P(rating=4)=1
- Prediction of item *B*: P(rating=5)=0.75, P(rating=1)=0.25
- E.g., use entropy of rating probability distribution
- ExpectedRating(A)=ExpectedRating(B)=4
  ➔Both items are equally "attractive"
- Entropy(A)=-1*log(1)=0, Entropy(B)=-0.75*log(0.75)-0.25*log(0.25)=0.81
  ➔More confident in recommending item A

Probability-based confidence

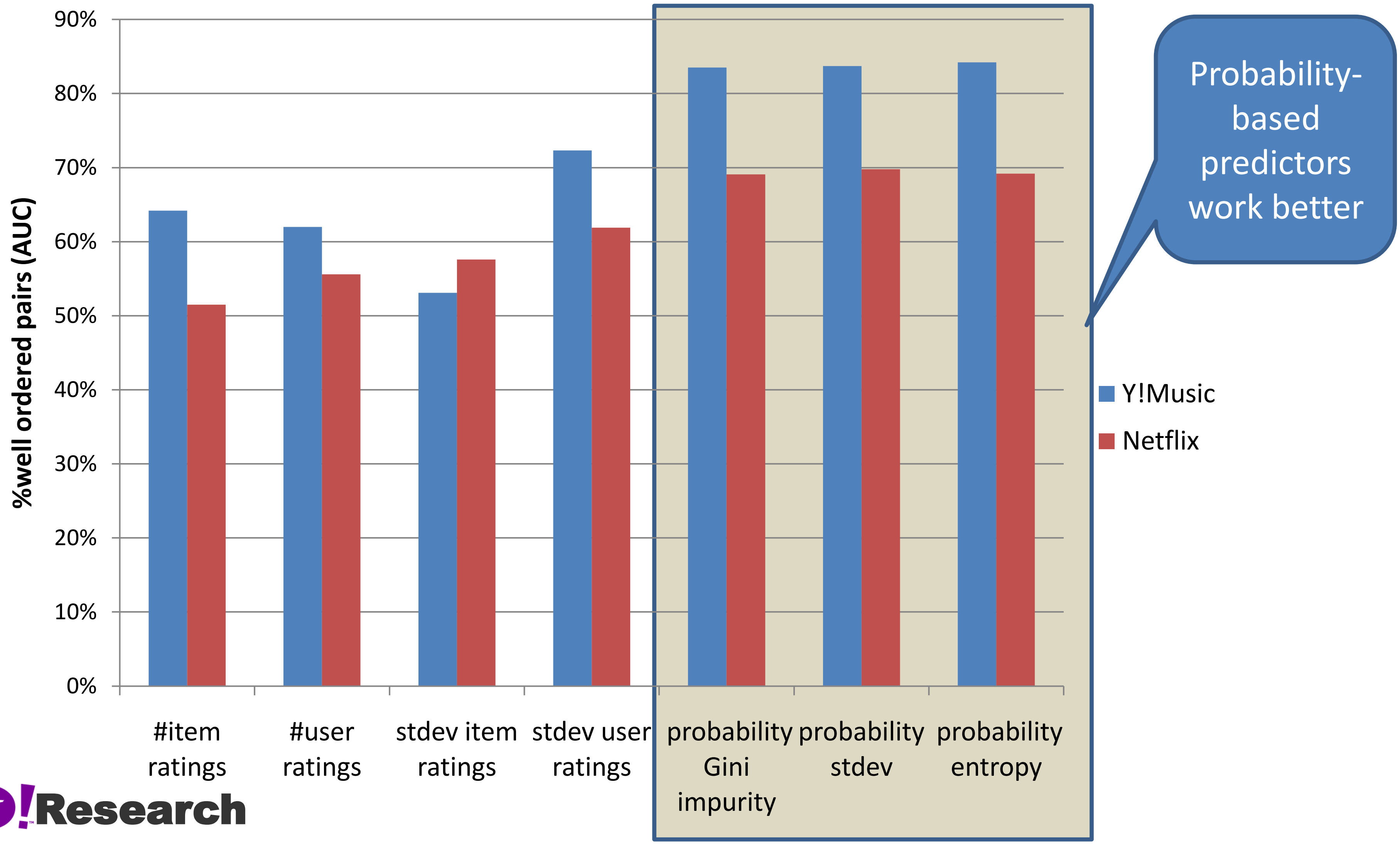
- Employ recommenders outputting full probability distribution of ratings
- Associate confidence with probability concentration
  - Use entropy, standard deviation, or Gini impurity of prob. distribution
- Personalized – depends on item and user combined
- Aware of predictor inner-working
  - Same (user,item)–pair can be difficult to one method, but easier to another

Evaluating confidence predictors

- We predicted ratings for Netflix and Yahoo! Music test sets
- We would like to identify "good" and "bad" predictions:
  - Good prediction – error<1, e.g. predict "4.2" instead of "5"
  - Bad prediction – error>1, e.g. predict "3.7" instead of "5"
- We expect: high confidence ➔ good prediction
- Confidence predictors act as classifiers separating "good" from "bad" predictions
- Quality of confidence predictor is measured by AUC (area under the ROC curve)

Evaluating confidence predictors

- Measure ability to separate error>1 by different confidence predictors:



Y! Research

Talk outline

- Recommender systems – a quick intro
  - Neighborhood methods
  - Matrix factorization methods
- Biases and temporal dynamics
- Bootstrapping a recommender – ratings elicitation
- Y!Answers – combining multiple kinds of attributes and feedback
- Interpretation of user feedback: binary, numeric, or ordinal?
- Estimating confidence in recommendations
- KDD-Cup'2011

kddcup.yahoo.com

Y! MUSIC

Y! Research

Yahoo! Music - Dataset

- 262,810,175 **Ratings**:
  <user id> <item id> <score> <date> <time>
  (Training: 252,800,275 Validation: 4,003,960 Test: 6,005,940)

- **Users**: 1,000,990 **Items**: 624,961
  Time period: 11 years

- **Taxonomy**:
  - Tracks: 507,172
  - Albums: 88,909
  - Artists: 27,888
  - Genres: 992

Y! MUSIC

kddcup.yahoo.com

KDD-Cup'11 –challenges

- Two tracks:
  - Track 1: minimize squared error on given ratings
  - Track 2: separate highly rated items from never rated items

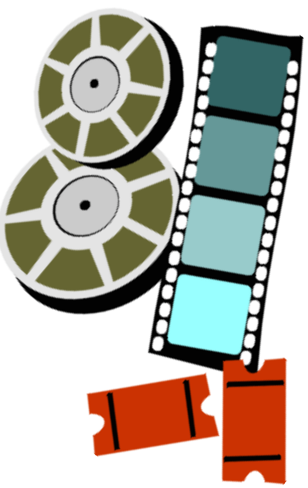
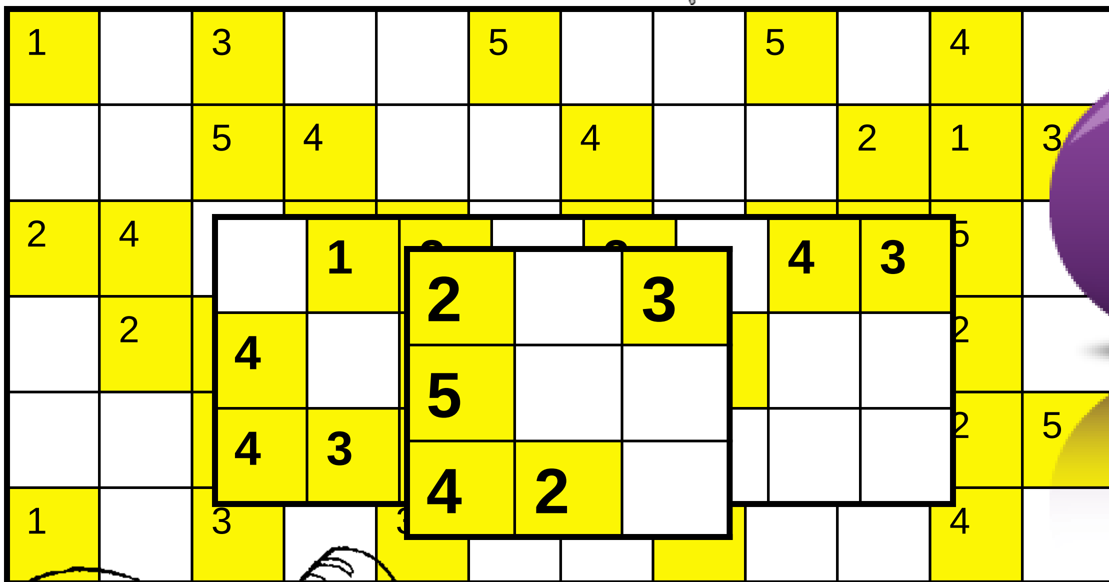
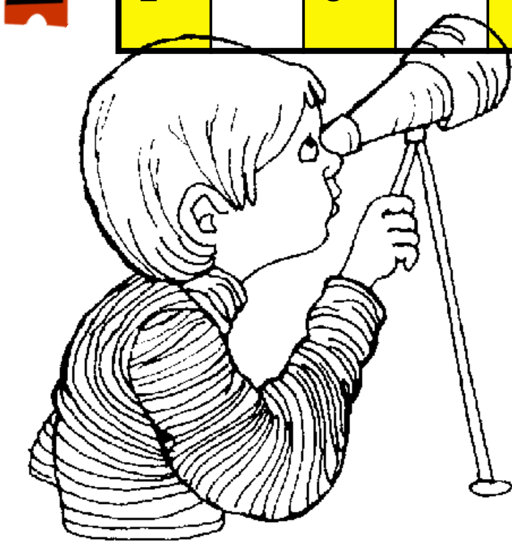
    ➔**Generalize models to items never rated by the users**
- Very large number of items (over 600K)
- Employ hierarchical relations (taxonomy) between items
- Accurate timestamps of ratings; facilitates session analysis

- *We would love to have you there!!*

kddcup.yahoo.com





Yehuda Koren

Yahoo! Research

yehuda@yahoo-inc.com