Recommending based on rating frequencies – Accurate enough?

Fatih Gedikli and Dietmar Jannach
TU Dortmund, Germany
dietmar.jannach@tu-dortmund.de

Background

- Collaborative filtering recommender systems
  - Recommendation of items based on community behavior
  - Assume that users who had similar tastes in the past, will have similar tastes in the future

CF algorithms

- Given
  – Users and item rating matrix
- Predict
  – Ratings for unseen items for active user

- Various algorithms proposed
  – Neighborhood-based approaches
  – Association rule mining
  – Probabilistic methods
  – Matrix factorization
  – …

Effectiveness of recommendations

- "Accuracy" the most common metric
  - Offline experimentation
    - MAE, RMSE
  - Measures deviation of predicted ratings from real (withheld) ratings

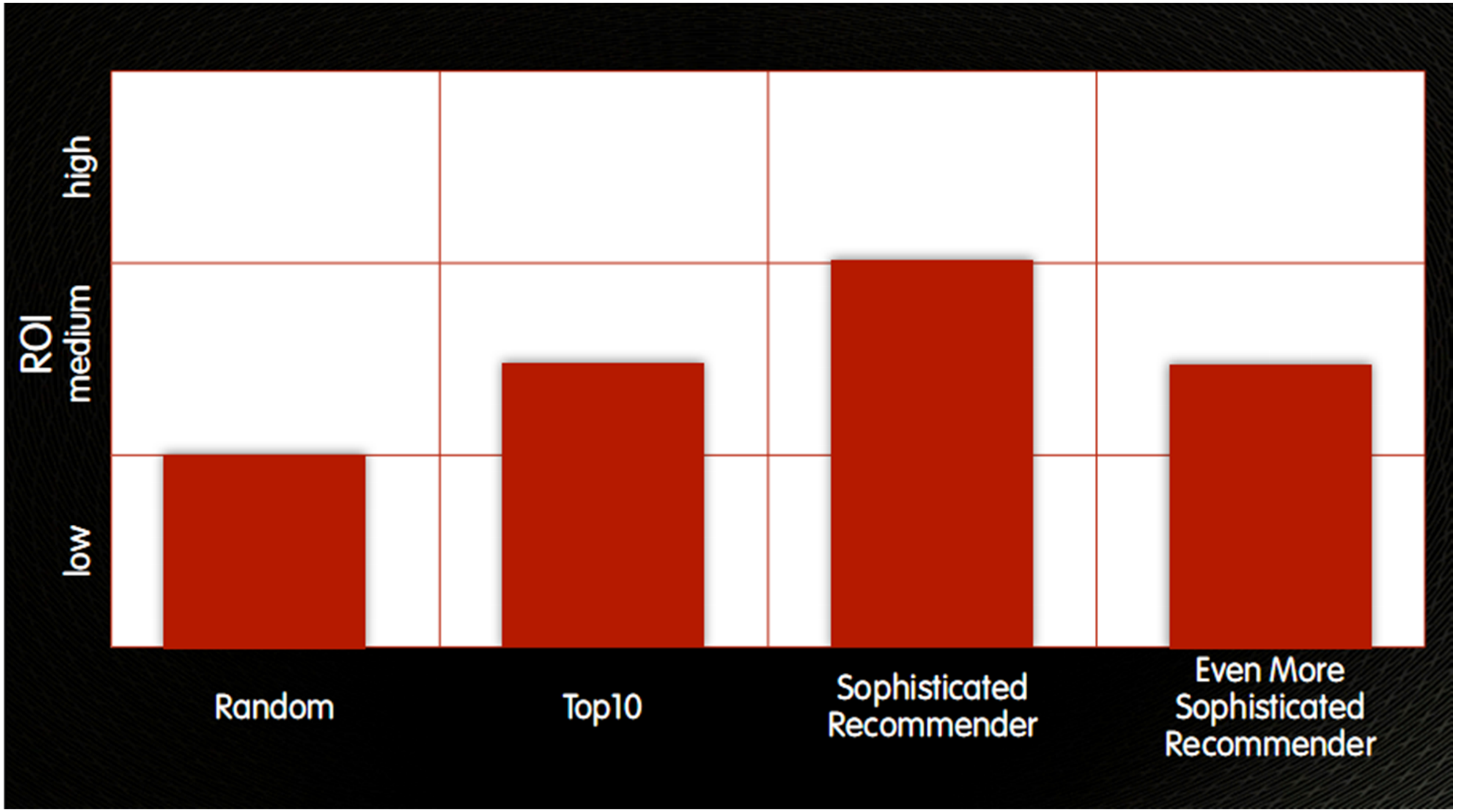
- Others are possible
  - Coverage, diversity, serendipity (novelty)
  - Navigation and purchase behavior

- Currently increased interest

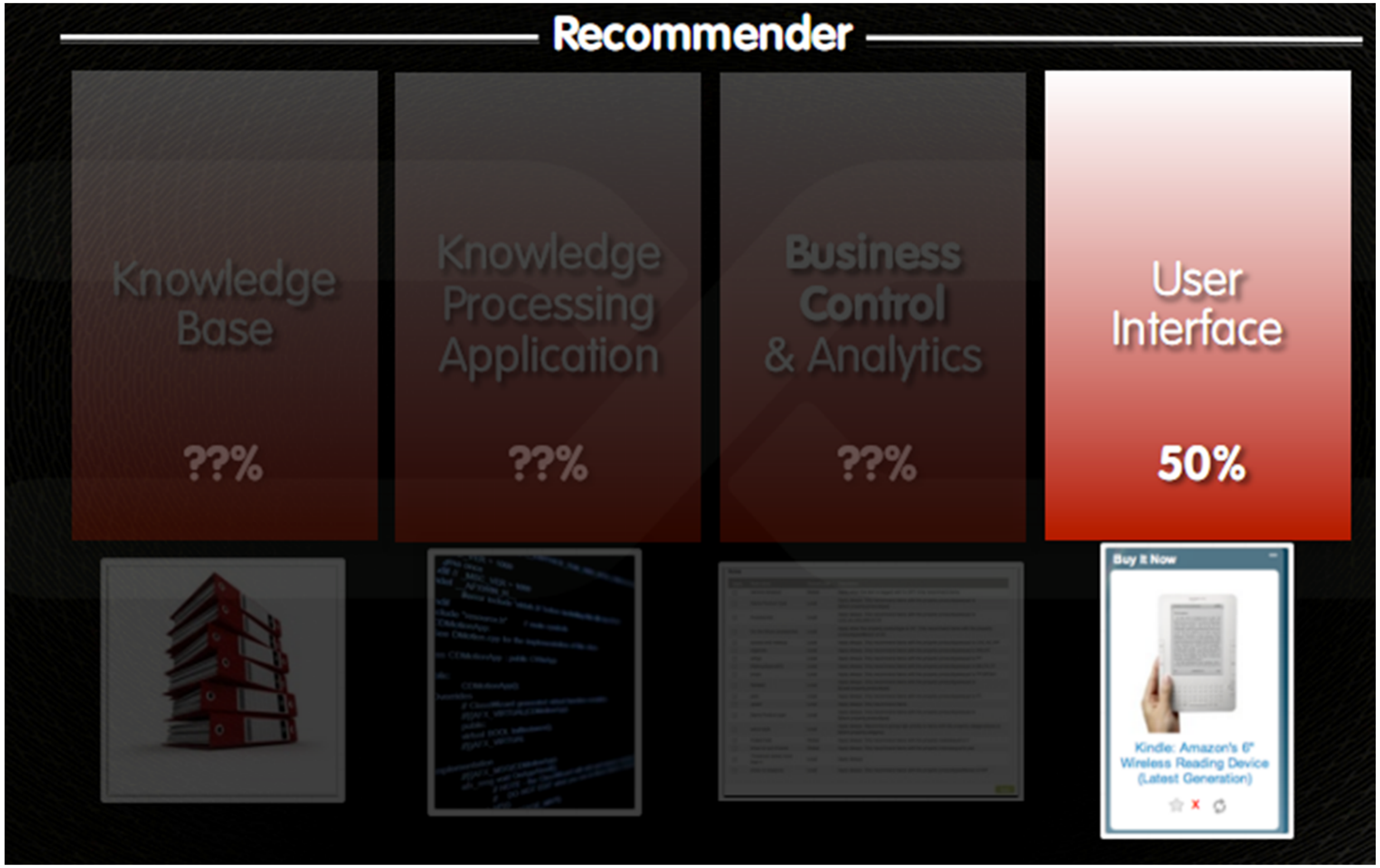
Other desirable features of an RS

- Implementation complexity
  - Easy to implement?
  - Availability of implementations in different languages?
- Offline-computation costs
  - Running times for large-scale systems?
  - Can the models be updated on the fly?
  - Parameter optimization costs?
- Efficiency at query time
- Explanations
- Accuracy (see next slide)

Industry wisdom (Francisco Martin, Strands)



More industry wisdom



Proposed method: RF-Rec

- Use a very simple prediction function
- Given:
  - For each possible rating value r, the number of times the active user U has used it.
  - For each possible rating value r, the number of times the target item has received this value from the community
- Predict:
  - The rating value that appeared most often for this user/item combination

More formally

$$pred(u,i) = \underset{r \in possibleRatings}{\arg\max} \Big((freqUser(u,r) + 1 + \mathbb{1}_{avg-user}(u,r)) * (freqItem(i,r) + 1 + \mathbb{1}_{avg-item}(i,r)) \Big)$$

- The indicator function returns 1
  - if the current rating is identical to the average rating
  - thus, serves as a tie-breaker and gives a light bias toward average ratings
- The 1 in the middle
  - Makes sure that factors are not zeroed out
  - In case a rating value was not used / given

Example

	I1	I2	I3	I4	I5	Average
Alice	1	1	?	5	4	2.75
U1	2		5	5	5	4.25
U2			1	1		1.00
U3		5	1	1	2	2.25
Average	1.50	3.00	2.33	3.00	3.67	

Rating value 1: (2+1+0)*(2+1+0) = 9

Rating value 2: (0+1+0)*(0+1+1) = 2

…

Rating value 5: (1+1+0)*(1+1+0) = 4

What is different?

	I1	I2	I3	I4	I5	Average
Alice	1	1	?	5	4	2.75
U1	2		5	5	5	4.25
U2			1	1		1.00
U3		5	1	1	2	2.25
Average	1.50	3.00	2.33	3.00	3.67	

- Both the ratings for I3 and the ones given by Alice are extreme
- Other approaches (kNN, SlopeOne) take **averages**
  - High variance in data can lead to decreased accuracy (Herlocker et al., 2000)
  - RF-Rec will also recommend extreme ratings
- Coverage:
  - Prediction possible if one item rating or one user rating is available.

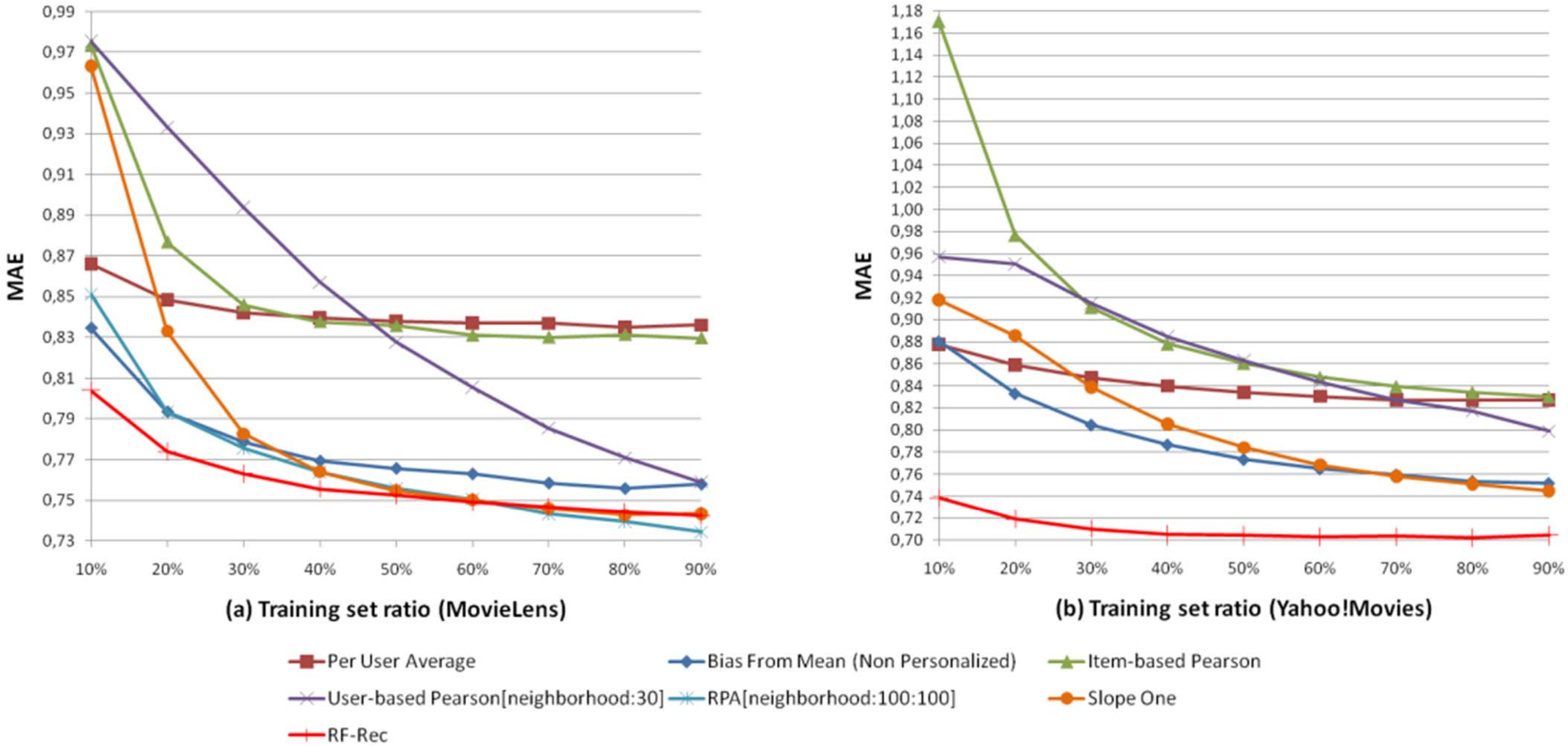
Experimental evaluation

- Three different data sets
  - MovieLens
    - 100.000 ratings from around 1.000 users on about 1.700 movies, sparsity 0,9369
  - Yahoo!Movies
    - 211.000 ratings by 7.600 users on 12.000 movies, sparsity 0,9976
  - BookCrossing (subset)
    - 100.000 ratings by 30.000 users on 37.400 books, sparsity 0,9999
- Variation of density level
  - From 10% to 90% (Train / test ratio)

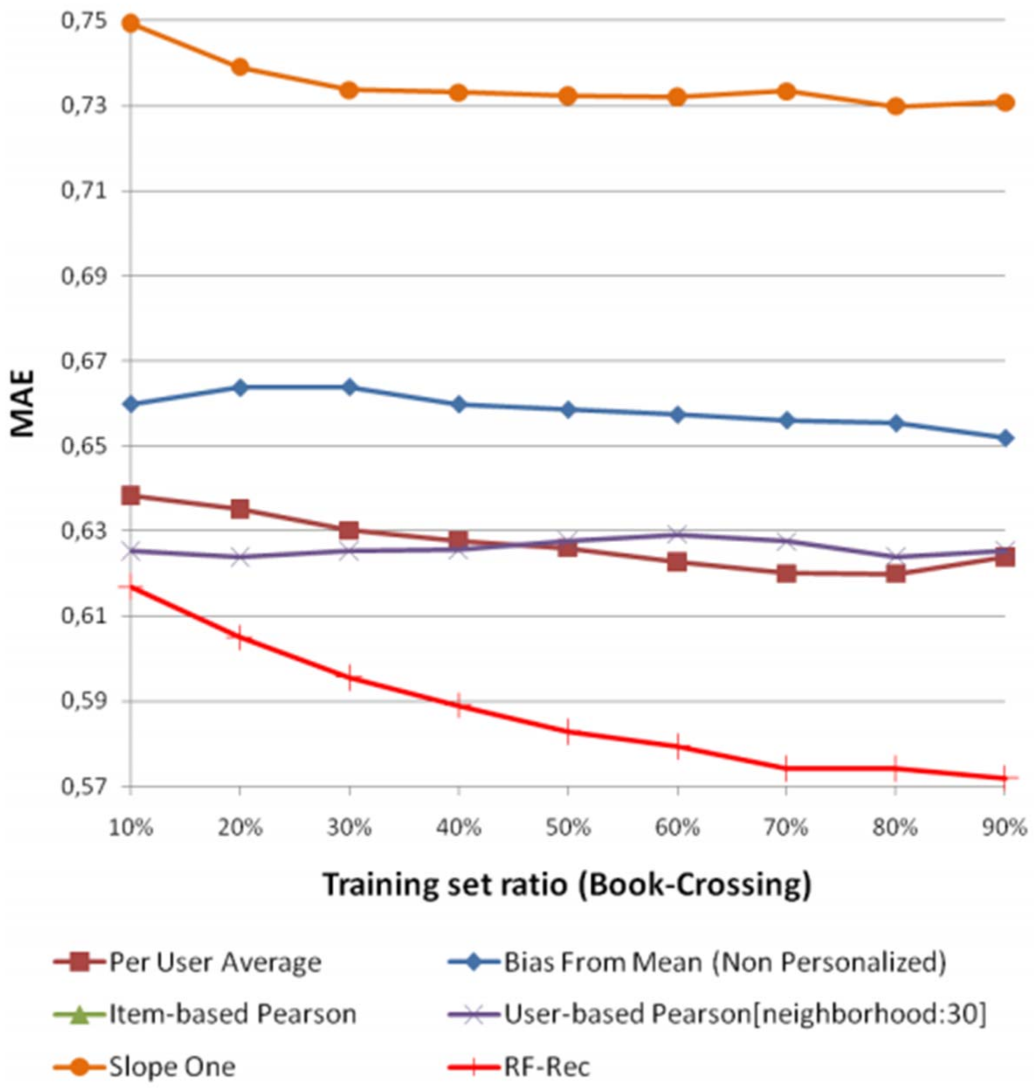
Experimental evaluation

- Evaluation metric
  - Mean Absolute Error
- Evaluated algorithms
  - User-based kNN with Pearson similarity and default voting; neighborhood size 30
  - Item-based kNN (Pearson correlation)
  - SlopeOne
  - BiasFromMean (Non-personalized)
  - RPA (Recursive prediction algorithm; Zhang & Pu, 2007)

Measurements



Measurements - BookCrossing



Observations

- Accuracy comparable or better to other methods
    - Except for costly RPA method for full MovieLens
- Accuracy even better for sparse data sets and low density levels
    - Item-based method: MAE 1.18 on BookCrossing
- Coverage 100%
  – MovieLens + kNN: Coverage slowly increases from 65% to 95%

Discussion of algorithm

- Implementation complexity is trivial
- Easy update when new data comes in
- Constant, minimal memory requirements
- No parameter optimization
- Generation of predictions very fast
- “Model-building” times
  - 500ms for the 1 million MovieLens dataset
  - 6 minutes for item-item (Mahout)
- Explanations?

Summary

- Accuracy of RF-Rec on a par with other (basic) algorithms
- Particularly good results for sparse data sets
    - Accuracy, coverage
- Result could help further re-focus RS research beyond accuracy
    – User interaction issues, marketing wisdom, psychology …

Future work

- Further evaluation
  - Other data sets
    - NetFlix
    - Other domains (tourism)
  - More sophisticated algorithms
    - Koren, 2009
    - Matrix factorization approaches
  - Variations of metrics
- Implementation of algorithm for different platforms and programming languages

Discussion

- Better quality metrics for recommender systems
- Repeatability of research in RS
  - Open source implementation provided by authors
  - Not common in the field
  - Evaluation scenario and parameter settings not described precisely in many papers