BITS Pilani
Pilani Campus

Information Retrieval

Abhishek
April 2020

innovate achieve lead

BITS Pilani
Pilani Campus

CS F469, Information Retrieval

Lecture topics: Recommender Systems

From Search to Recommendation

- "The Web is leaving the era of search and entering one of discovery. What's the difference?
- Search is what you do when you're looking for something.
- Discovery is when something wonderful that you didn't know existed, or didn't know how to ask for, finds you." –CNN Money, "The race to create a 'smart' Google.

Customer Who Bought This Item Also Bought

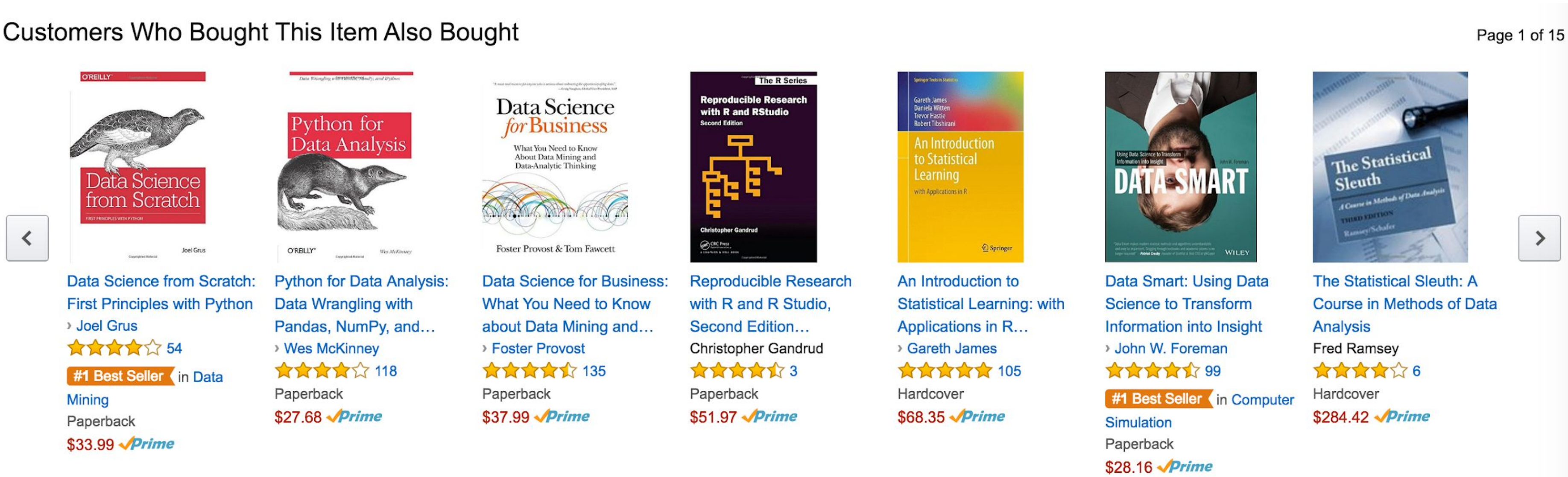


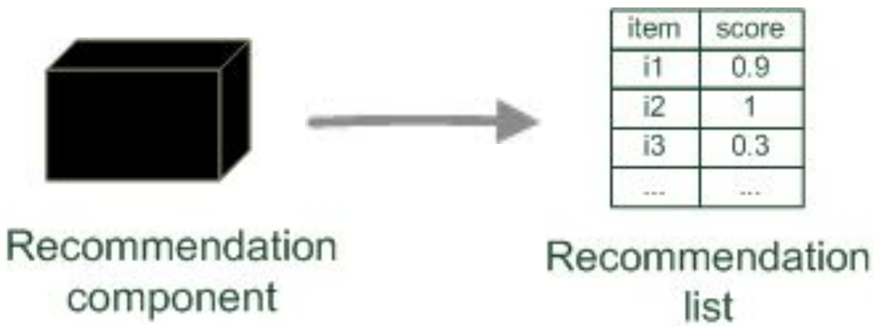
Image Source: https://jessesw.com/Rec-System/

Recommender Systems all almost everywhere

- E-commerce websites
- Online advertisements
- Social Media Platforms
- Dating Platforms
- Jobs

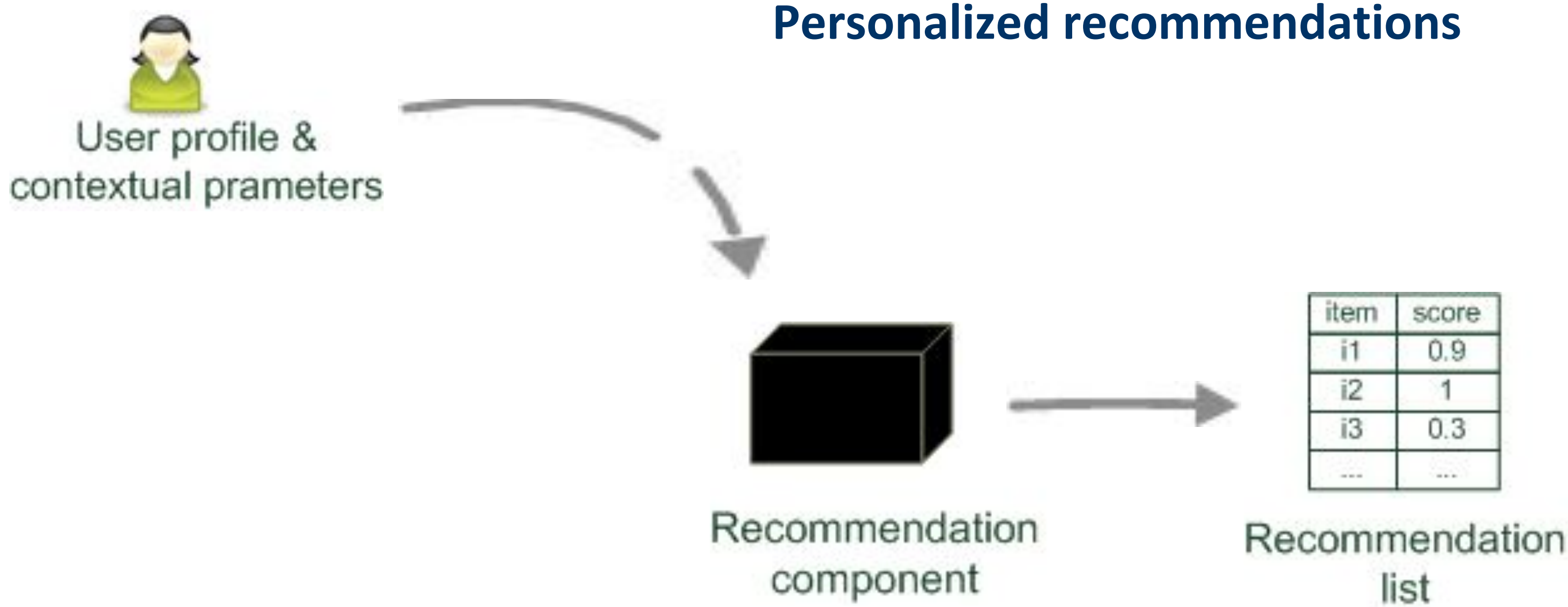
Paradigms of recommender systems

Recommender systems reduce information overload by estimating relevance



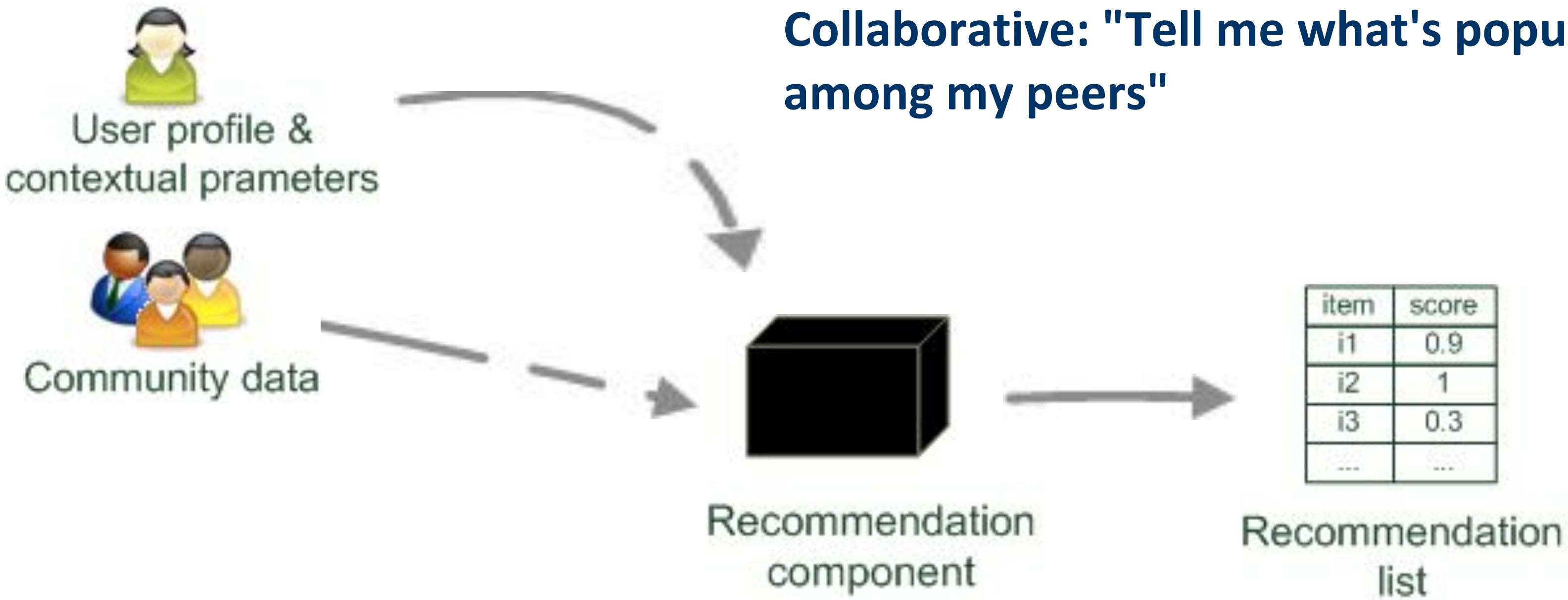
Paradigms of recommender systems

Personalized recommendations



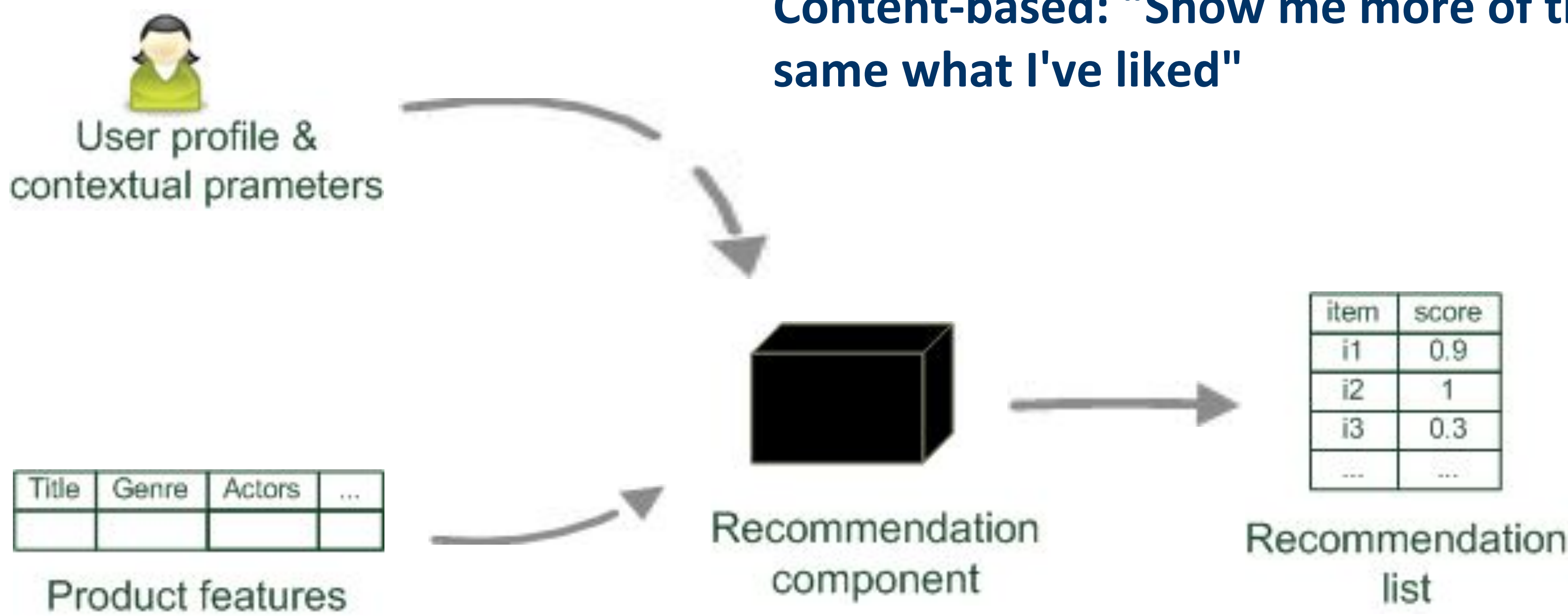
Paradigms of recommender systems

Collaborative: "Tell me what's popular among my peers"



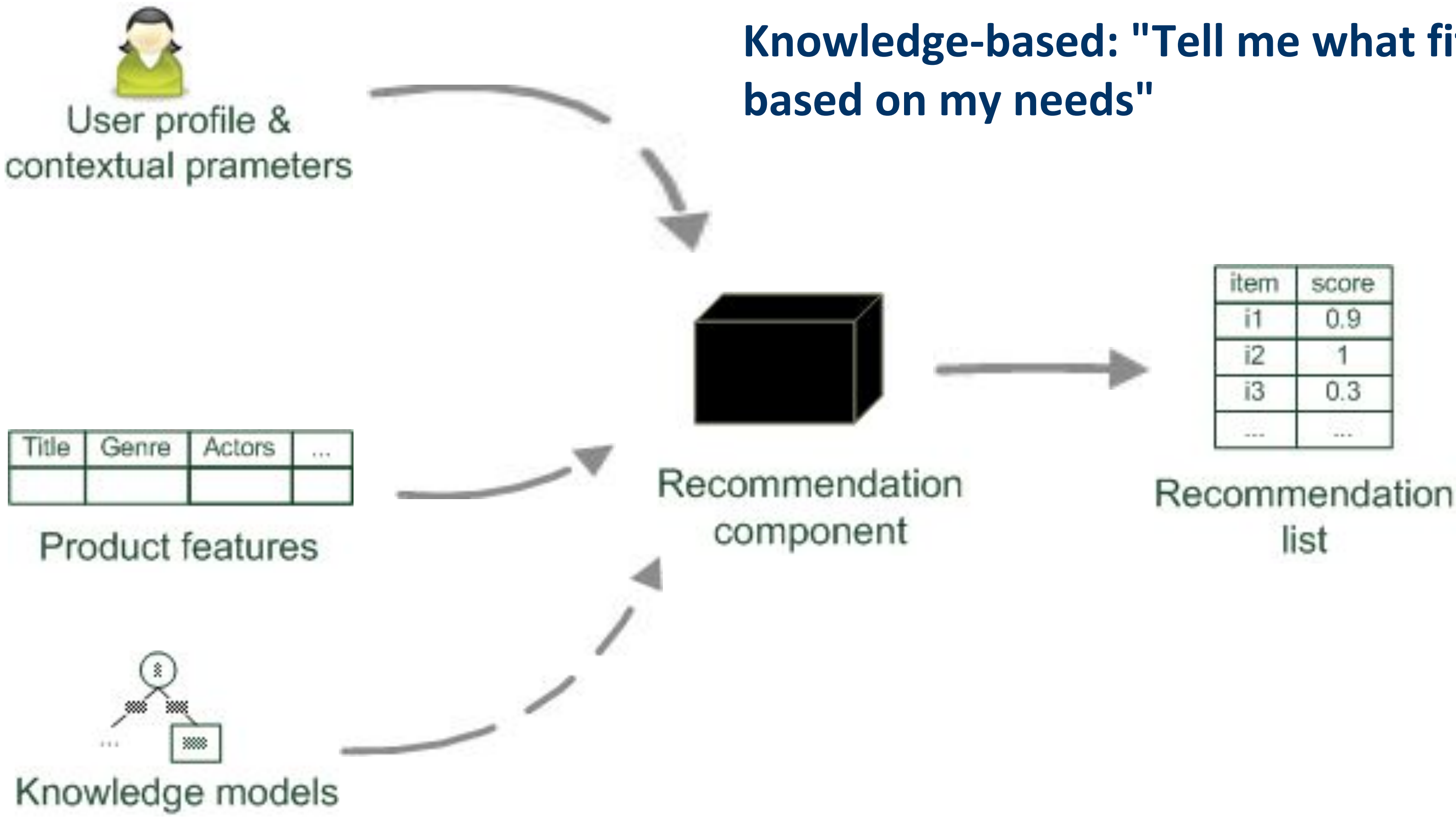
Paradigms of recommender systems

Content-based: "Show me more of the same what I've liked"



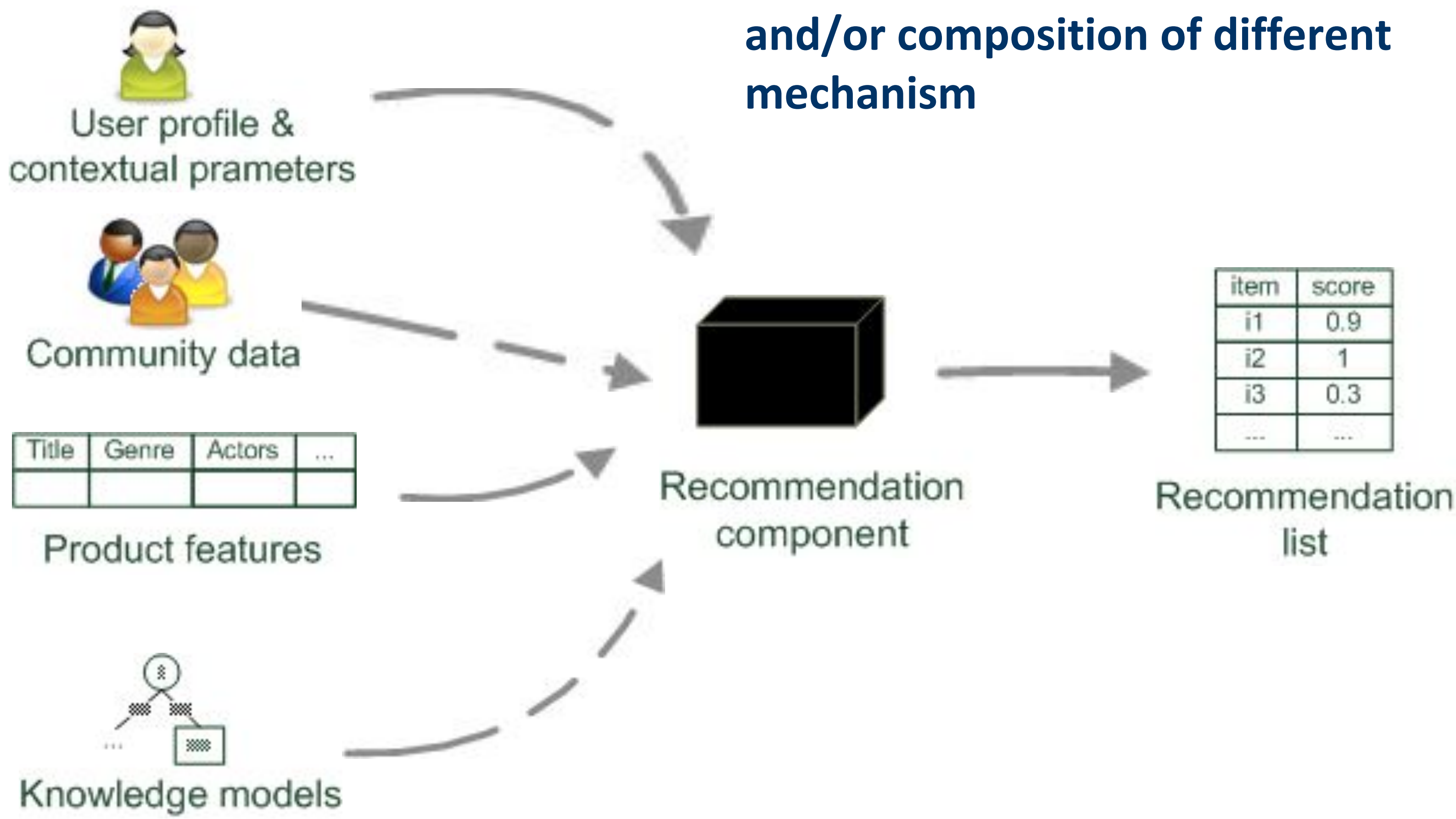
Paradigms of recommender systems

Knowledge-based: "Tell me what fits based on my needs"



Paradigms of recommender systems

Hybrid: combinations of various inputs and/or composition of different mechanism



Collaborative Filtering

- The most prominent approach to generate recommendations
  - used by large, commercial e-commerce sites.
  - well-understood, various algorithms and variations exist.
  - applicable in many domains (book, movies, DVDs, ..)
- **Approach**
  - use the "wisdom of the crowd" to recommend items
- **Basic assumption and idea**
  - Users give ratings to catalog items (implicitly or explicitly)
  - Customers who had similar tastes in the past, will have similar tastes in the future

Examples of its use include Amazon, iTunes, Netflix, LastFM, StumbleUpon, and Delicious.

Pure CF Approaches

- **Input**
  - Only a matrix of given user–item ratings
- **Output types**
  - A (numerical) prediction indicating to what degree the current user will like or dislike certain items
  - A top-N list of recommended items

User-based nearest-neighbor collaborative filtering (1)

The basic technique

- Given an "active user" (Alice) and an item **I** not yet seen by Alice
- The goal is to estimate Alice's rating for this item, e.g., by
  - find a set of users (peers) who liked the same items as Alice in the past **and** who have rated item **I**
  - use, e.g. the average of their ratings to predict, if Alice will like item **I**
  - do this for all items Alice has not seen and recommend the best-rated

	Item 1	Item 2	Item 3	Item 4	Item 5
Alice	5	3	4	4	?
User 1	3	1	2	3	3
User 2	4	3	4	3	5
User 3	3	3	1	5	4
User 4	1	5	5	2	1

User-based nearest-neighbor collaborative filtering (2)

- **Some first questions**
  - How do we measure similarity?
  - How many neighbors should we consider?
  - How do we generate a prediction from the neighbors' ratings?

	Item 1	Item 2	Item 3	Item 4	Item 5
Alice	5	3	4	4	?
User 1	3	1	2	3	3
User 2	4	3	4	3	5
User 3	3	3	1	5	4
User 4	1	5	5	2	1

Measuring user similarity (1)

- A popular similarity measure in user-based CF: **Pearson correlation**
  - a, b : users
  - $r_{a,p}$: rating of user a for item p
  - P : set of items, rated both by a and b
  - Possible similarity values between -1 and 1

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \overline{r_a})(r_{b,p} - \overline{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \overline{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \overline{r_b})^2}}$$

Measuring user similarity example

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \overline{r_a})(r_{b,p} - \overline{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \overline{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \overline{r_b})^2}}$$

	Item 1	Item 2	Item 3	Item 4	Item 5
Alice	5	3	4	4	?
User 1	3	1	2	3	3
User 2	4	3	4	3	5
User 3	3	3	1	5	4
User 4	1	5	5	2	1

sim = 0.85

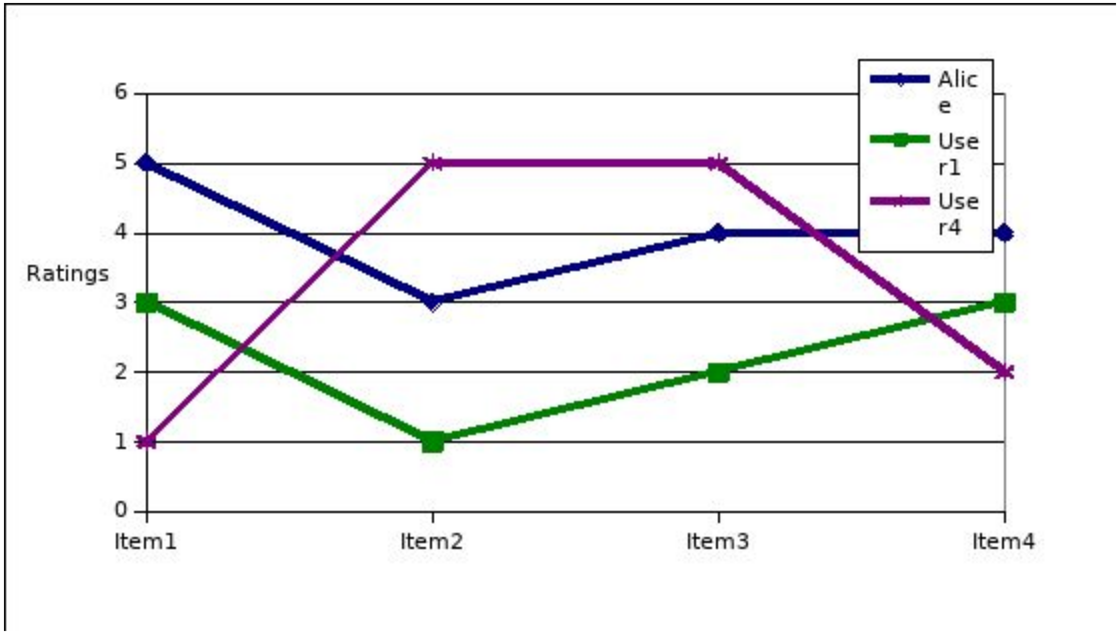
sim = 0.00

sim = 0.70

sim = -0.79

Pearson correlation

- Takes differences in rating behavior into account



- Works well in usual domains, compared with alternative measures
  - such as cosine similarity

Making predictions

- A common prediction function:

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

- Calculate, whether the neighbors' ratings for the unseen item are higher or lower than their average.
- Combine the rating differences – use the similarity with as a weight
- Add/subtract the neighbors' bias from the active user's average and use this as a prediction

Prediction Example

$$4 + 1/(0.85 + 0.7) * (0.85 * (3 - 2.4) + 0.70 * (5 - 3.8)) = 4.87$$

	Item 1	Item 2	Item 3	Item 4	Item 5	
Alice	5	3	4	4	?	
User 1	3	1	2	3	3	sim = 0.85
User 2	4	3	4	3	5	sim = 0.00
User 3	3	3	1	5	4	sim = 0.70
User 4	1	5	5	2	1	sim = -0.79

Better Similarity, weighting metrics and neighbour selection

- Not all neighbour ratings might be equally "valuable"
  - Agreement on commonly liked items is not so informative as agreement on controversial items
  - Possible solution: Give more weight to items that have a higher variance. (Breese et al. 1998)
- Value of number of co-rated items
  - Use "significance weighting", by e.g., linearly reducing the weight when the number of co-rated items is low . (Herlocker et al. 1999, 2002)
- Case amplification
  - Intuition: Give more weight to "very similar" neighbors, i.e., where the similarity value is close to 1. (Breese et al. 1998)
- Neighborhood selection
  - Use similarity threshold or fixed number of neighbors. (Anand and Mobasher 2005)

References

- Recommender Systems: An Introduction
  - http://www.recommenderbook.net/
  - Slides from Introduction chapter and CF chapter.

Papers:

- J. Herlocker, J. A. Konstan, and J. Riedl, *An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms*, Information Retrieval 5 (2002), no. 4, 287–310.
- J. L. Herlocker, J. A. Konstan, et al., *An Algorithmic Framework for Performing Collaborative Filtering*, Proceedings of the 22nd Annual International ACM SIGIR Conference, ACM Press, 1999, pp. 230–237.
- J. S. Breese, D. Heckerman, and C. M. Kadie, *Empirical analysis of predictive algorithms for collaborative filtering*, Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (Madison, WI) (Gregory F. Cooper and Serafín Moral, eds.), Morgan Kaufmann, 1998, pp. 43–52.
- S. S. Anand and B. Mobasher, *Intelligent techniques for web personalization*, Lecture Notes in Computer Science, vol. 3169, Springer, Acapulco, Mexico, 2005, pp. 1–36.

Item-based collaborative filtering

- Basic idea:
  - Use the similarity between items (and not users) to make predictions
- Example:
  - Look for items that are similar to Item 5
  - Take Alice's ratings for these items to predict the rating for Item 5

	Item 1	Item 2	Item 3	Item 4	Item 5
Alice	5	3	4	4	?
User 1	3	1	2	3	3
User 2	4	3	4	3	5
User 3	3	3	1	5	4
User 4	1	5	5	2	1

The cosine similarity measure

- **Produces better results in item-to-item filtering**
  - For some datasets, no consistent picture in literature
- **Ratings are seen as vector in n-dimensional space**
- **Similarity is calculated based on the angle between the vectors**

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

	Item 1	Item 2	Item 3	Item 4	Item 5
Alice	5	3	4	4	?
User 1	3	1	2	3	3
User 2	4	3	4	3	5
User 3	3	3	1	5	4
User 4	1	5	5	2	1

$$sim(I5, I1) = \frac{3*3+5*4+4*3+1*1}{\sqrt{3^2+5^2+4^2+1^2} * \sqrt{3^2+4^2+3^2+1^2}} = 0.99$$

Adjusted cosine similarity

- Take average user ratings into account, transform the original ratings

$$sim(a,b) = \frac{\sum_{u \in U}(r_{u,a} - \overline{r_u})(r_{u,b} - \overline{r_u})}{\sqrt{\sum_{u \in U}(r_{u,a} - \overline{r_u})^2}\sqrt{\sum_{u \in U}(r_{u,b} - \overline{r_u})^2}}$$

	Item1	Item2	Item3	Item4	Item5
Alice	1.00	−1.00	0.00	0.00	?
User1	0.60	−1.40	−0.40	0.60	0.60
User2	0.20	−0.80	0.20	−0.80	1.20
User3	−0.20	−0.20	−2.20	2.80	0.80
User4	−1.80	2.20	2.20	−0.80	−1.80

$$\frac{0.6 * 0.6 + 0.2 * 1.2 + (-0.2) * 0.80 + (-1.8) * (-1.8)}{\sqrt{(0.6^2 + 0.2^2 + (-0.2)^2 + (-1.8)^2} * \sqrt{0.6^2 + 1.2^2 + 0.8^2 + (-1.8)^2}} = 0.80$$

Making predictions

- A common prediction function:

$$pred(u,p) = \frac{\sum_{i \in ratedItems(u)} sim(i,p) * r_{u,i}}{\sum_{i \in ratedItems(a)} sim(i,p)}$$

- Neighborhood size is typically also limited to a specific size
- Not all neighbors are taken into account for the prediction
- An analysis of the MovieLens dataset indicates that "in most real-world situations, a neighborhood of 20 to 50 neighbors seems reasonable" (Herlocker et al. 2002)

Pre-processing for item-based filtering

- Item-based filtering does not solve the scalability problem itself
- **Pre-processing approach by Amazon.com (Linden et al. 2003)**
  - Calculate all pairwise item similarities in advance
  - The neighborhood to be used at run-time is typically rather small, because only items are taken into account which the user has rated
  - Item similarities are supposed to be more stable than user similarities
- **Memory requirements**
  - Up to N^2 pairwise similarities to be memorized (N = number of items) in theory
  - In practice, this is significantly lower (items with no co-ratings)
  - Further reductions possible

References

- Recommender Systems: An Introduction
  - http://www.recommenderbook.net/
  - Slides from Introduction chapter and CF chapter.

Papers:

- G. Linden, B. Smith, and J. York, Amazon.com recommendations: item-to-item collaborative filtering, Internet Computing, IEEE 7 (2003), no. 1, 76–80.

Data sparsity problems

- **Cold start problem**
  - How to recommend new items? What to recommend to new users?
- **Straightforward approaches**
  - Ask/force users to rate a set of items
  - Use another method (e.g., content-based, demographic or simply non-personalized) in the initial phase
- **Alternatives**
  - Use better algorithms (beyond nearest-neighbor approaches)
  - Example:
    - In nearest-neighbor approaches, the set of sufficiently similar neighbors might be too small to make good predictions
    - Assume "transitivity" of neighborhoods

Example algorithms for sparse datasets

- **Recursive CF (Zhang and Pu 2007)**
  - Assume there is a very close neighbor **n** of **u** who however has not rated the target item **i** yet.
- **Idea:**
  - Apply CF-method recursively and predict a rating for item for the neighbor
  - Use this predicted rating instead of the rating of a more distant direct neighbor

	Item 1	Item 2	Item 3	Item 4	Item 5
Alice	5	3	4	4	?
User 1	3	1	2	3	?
User 2	4	3	4	3	5
User 3	3	3	1	5	4
User 4	1	5	5	2	1

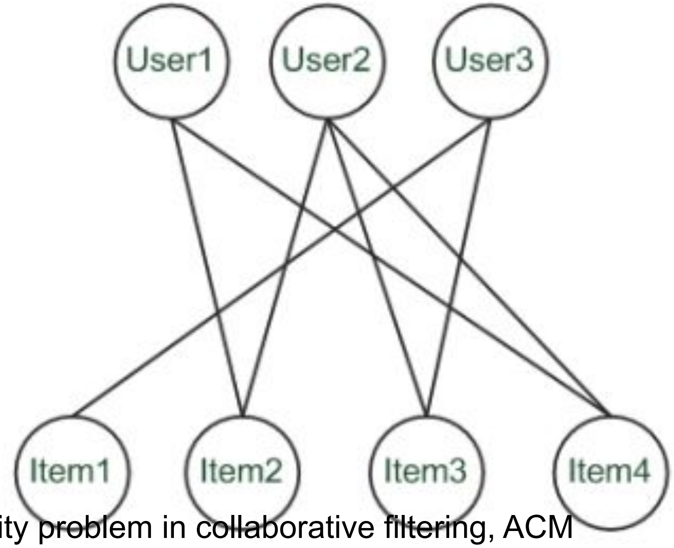
sim = 0.85

Predict rating for user 1

[Zhang and Pu 2007] A recursive prediction algorithm for collaborative filtering recommender systems, RecSys ’07

Graph-based methods (1)

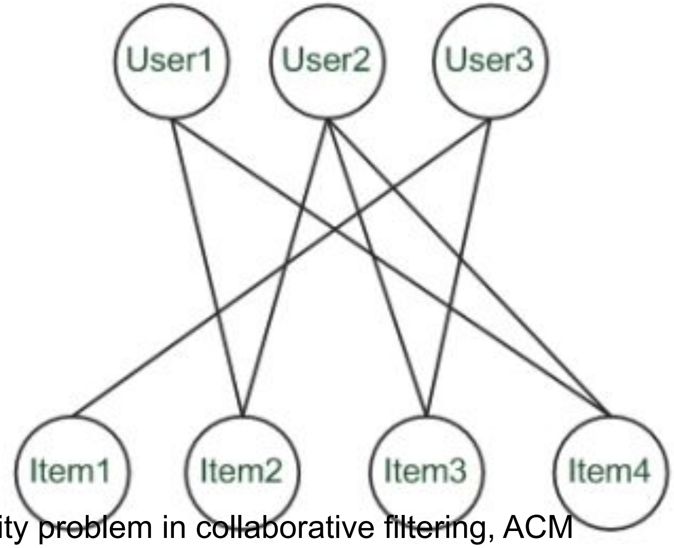
- **"Spreading activation" (Huang et al. 2004)**
  - Exploit the supposed "transitivity" of customer tastes and thereby augment the matrix with additional information
  - Assume that we are looking for a recommendation for User 1
  - When using a standard CF approach, User 2 will be considered a peer for User 1 because they both bought Item 2 and Item 4
  - Thus, Item 3 will be recommended to User 1 because the nearest neighbor, User 2, also bought or liked it



[Huang et al. 2004] Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering, ACM Transactions on Information Systems, 2004

Graph-based methods (1)

- **"Spreading activation" (Huang et al. 2004)**
  - Idea: Use paths of lengths > 3 to recommend items
  - Length 3: Recommend Item 3 to User 1
  - Length 5: Item 1 also recommendable



[Huang et al. 2004] Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering, ACM Transactions on Information Systems, 2004

More model-based approaches

- **Plethora of different techniques proposed in the last years, e.g.,**
  - Matrix factorization techniques, statistics
    - singular value decomposition, principal component analysis
  - Association rule mining
    - compare: shopping basket analysis
  - Probabilistic models
    - clustering models, Bayesian networks, probabilistic Latent Semantic Analysis
  - Various other machine learning approaches

Association rule mining

- Commonly used for shopping behavior analysis
    - aims at detection of rules such as

      "If a customer purchases baby food then the customer also buys diapers in 70% of the cases"

- Association rule mining algorithms
    - can detect rules of the form A → B (e.g., baby food → diapers) from a set of sales transactions $D = \{t_1, t_2, \ldots t_n\}$
    - measure of quality: support, confidence
    - let X is a set of items, and **C**(X) denotes that in how many transactions (t_i), X is present.
    - support = **C**(X U Y) / |D| , confidence = **C**(X U Y) / **C**(X)

Recommendation based on Association Rule Mining

- Simplest approach
  - transform 5-point ratings into binary ratings (1 = above user average)

	Item 1	Item 2	Item 3	Item 4	Item 5
Alice	1	0	0	0	?
User 1	1	0	1	0	1
User 2	1	0	1	0	1
User 3	0	0	0	1	1
User 4	0	1	1	0	0

- Mine rules such as
  - Item 1 → Item 5
    - support (2/4) confidence (2/2) (without Alice)

- Make recommendations for Alice (basic method)
  - Determine "relevant" rules based on Alice's transactions

    (the above rule will be relevant as Alice bought Item 1)
  - Determine items not already bought by Alice
  - Sort the items based on the rules' confidence confidence values

Collaborative Filtering Issues

- **Pros:**
  - well-understood, works well in some domains, no knowledge engineering required
- **Cons:**
  - requires user community, sparsity problems, no integration of other knowledge sources, no explanation of results
- **What is the best CF method?**
  - In which situation and which domain? Inconsistent findings; always the same domains and data sets; differences between methods are often very small.
- **How to evaluate the prediction quality?**
  - MAE / RMSE: What does an MAE of 0.7 actually mean?
    - Serendipity (novelty and surprising effect of recommendations)
    - Not yet fully understood

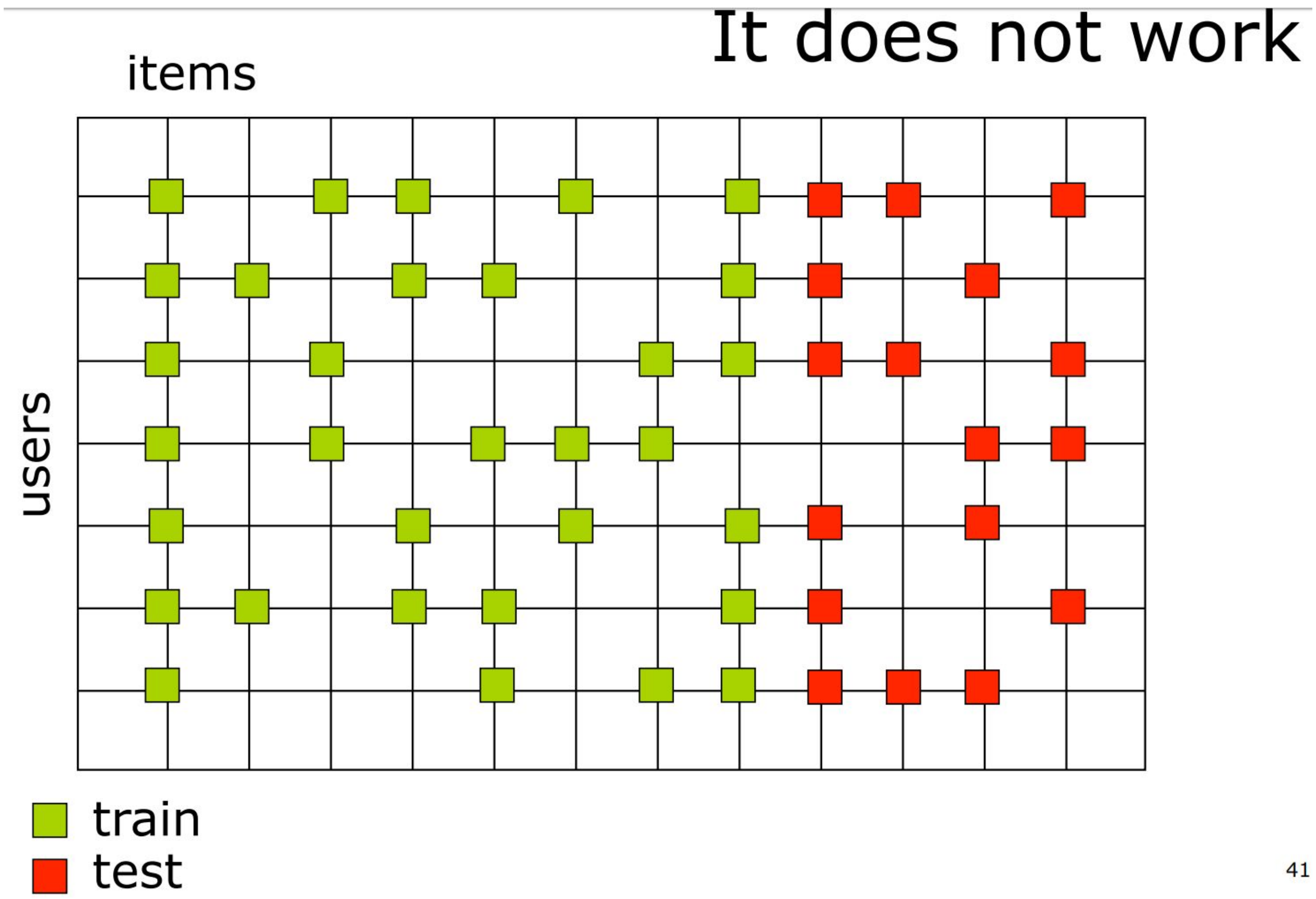
Evaluating Recommender Systems

Traditional Offline Experimentations

- **Split** the available data (so you need to collect data first!), i.e., the user-item ratings into two sets: **training** and **test.**
- **Build a model** on the training data
  - For instance, in a nearest neighbor CF simply put the ratings in the training in a separate set.
- **Compare the predicted …**
  - **rating on each test item** (user-item combination) with the true rating stored in the test set
  - **recommendations** with the really good recommendations (what are they?)
  - **ranking** with the correct ranking (what is this?)
- You need a **metric** to compare the **predicted** rating (or recommendation or ranking) with the **true** rating (or recommendation or ranking).

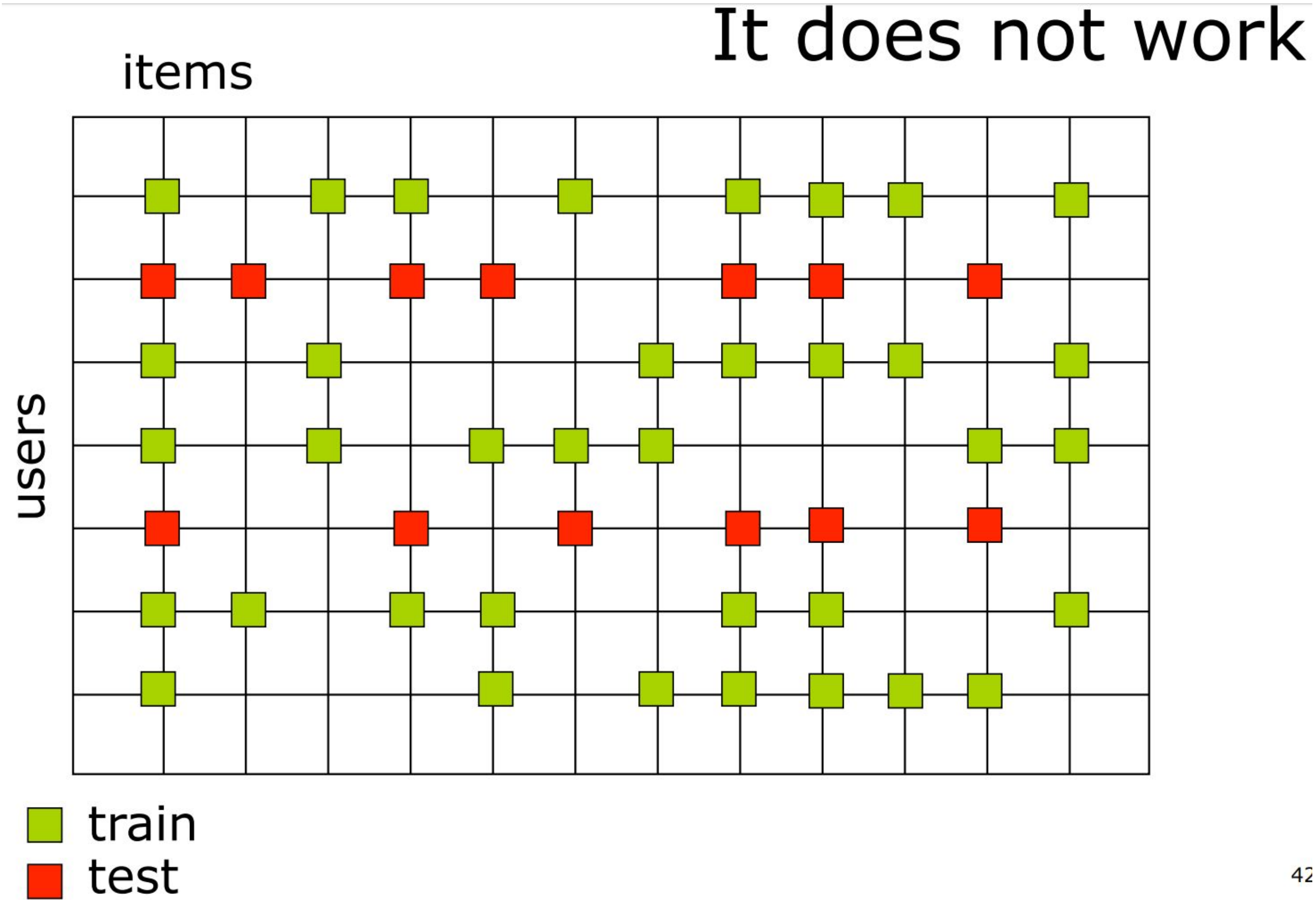
Slide from **Francesco Ricci** lecture slides available at: http://www.inf.unibz.it/~ricci/ISR/#reading_material

Splitting the data



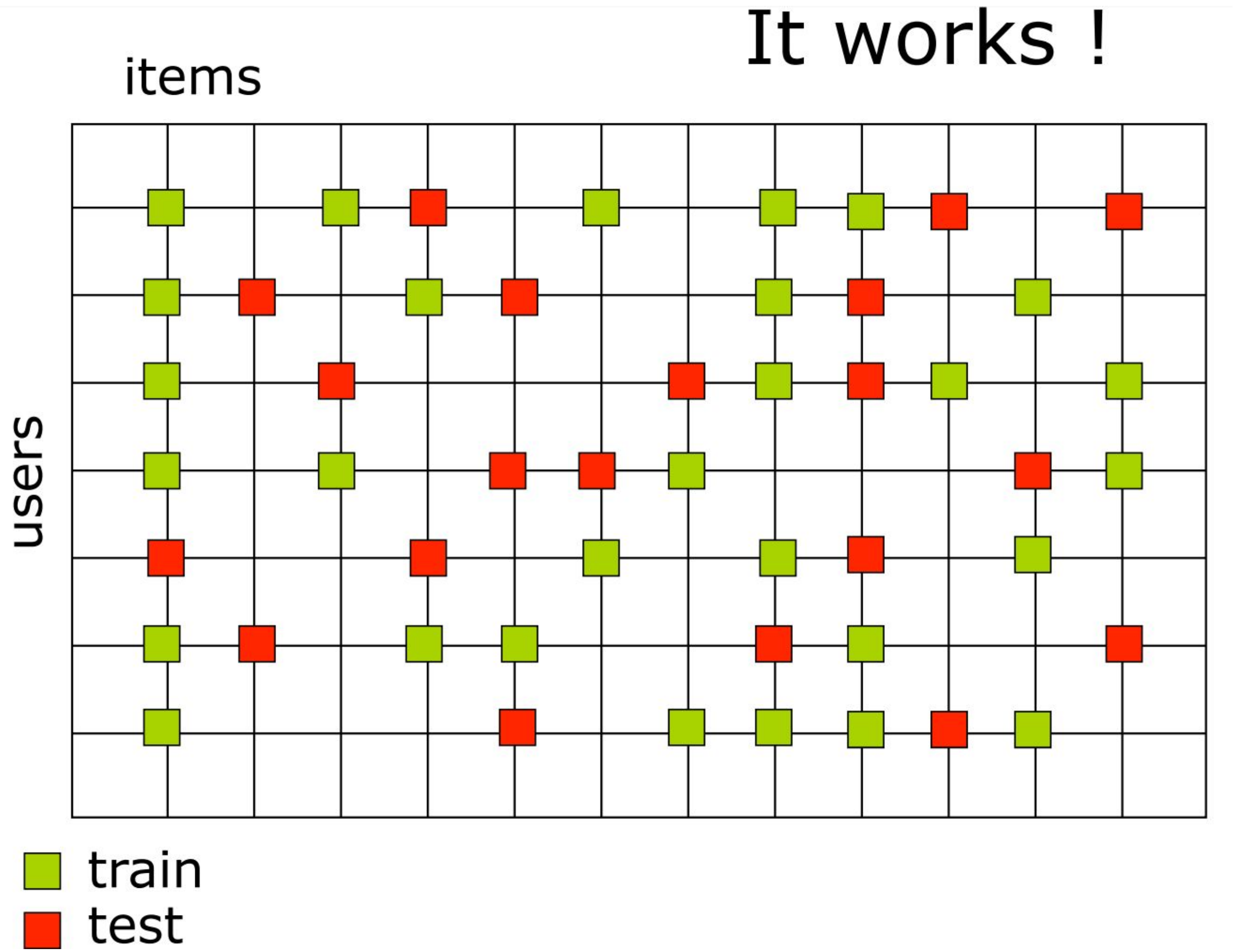
Slide from **Francesco Ricci** lecture slides available at: http://www.inf.unibz.it/~ricci/ISR/#reading_material

Splitting the data



Slide from **Francesco Ricci** lecture slides available at: http://www.inf.unibz.it/~ricci/ISR/#reading_material

Splitting the data



Slide from **[Francesco Ricci](#)** lecture slides available at: http://www.inf.unibz.it/~ricci/ISR/#reading_material

Comparing Values

- Measure **how close** the predicted ratings are to the true user ratings (for all the ratings in the test set)
- **Mean Absolute Error (MAE)** computes the deviation between predicted ratings and actual ratings

$$MAE = \frac{1}{n}\sum_{i=1}^{n} | p_i - r_i |$$

- **Root Mean Square Error (RMSE)** is similar to MAE, but places more emphasis on larger deviation

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (p_i - r_i)^2}$$

Comparing Recommendations: Precision and Recall

- **Recommendation is viewed as information retrieval task:**
  - Retrieve (recommend) all items which are predicted to be “good”.
- **Precision: a measure of exactness, determines the fraction of relevant items retrieved out of all items retrieved.**
  - E.g. the proportion of recommended movies that are actually good.

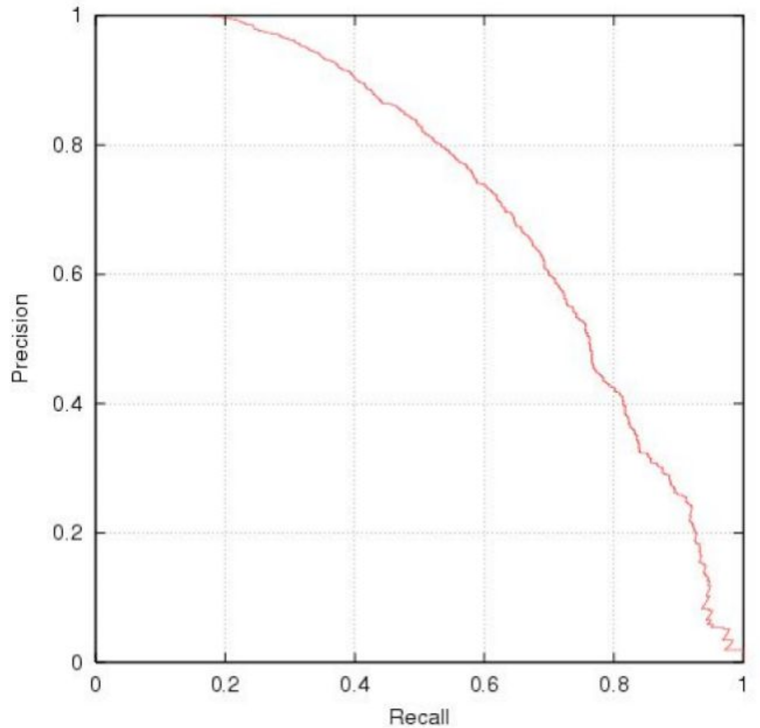
$$Precision = \frac{tp}{tp + fp} = \frac{|good\ movies\ recommended|}{|\text{all recommendations}|}$$

- **Recall: a measure of completeness, determines the fraction of relevant items retrieved out of all relevant items.**
  - E.g. the proportion of all good movies recommended

$$Recall = \frac{tp}{tp + fn} = \frac{|good\ movies\ recommended|}{|all\ good\ movies|}$$

Precision vs. Recall

- E.g. typically when a recommender system is tuned to increase precision, recall decreases as a result (or vice versa)



Metrics: Rank position matters

- For a user:

Actually good
Item 237
Item 899

Recommended (predicted as good)
Item 345
Item 237
Item 187

- Rank metrics extend recall and precision to take the positions of correct items in a ranked list into account.
  - Relevant items are more useful when they appear earlier in the recommendation list.
  - Particularly important in recommender systems as lower ranked items may be overlooked by users.

Metrics: Normalized Discounted Cumulative Gain

- Discounted cumulative gain (DCG)
    - Logarithmic reduction factor

$$DCG_{pos} = rel_1 + \sum_{i=2}^{pos} \frac{rel_i}{\log_2 i}$$

Where:

- pos denotes the position up to which relevance is accumulated
- rel_i returns the relevance of recommendation at position **i**

Normalized discounted cumulative gain (nDCG)

- Divide by the ideal recommendations DCG
- Normalized to the interval [0..1]

There are also other ranking metrics such as **Rank score**, **Liftindex, …**

Offline experimentation example

- **Netflix competition**
    - Web-based movie rental
    - Prize of $1,000,000 for accuracy improvement (RMSE) of 10% compared to own Cinematch system.

- **Historical dataset**
    - ~480K users rated ~18K movies on a scale of 1 to 5 ~100M ratings
    - Last 9 ratings/user withheld
    - Probe set – for teams for evaluation
    - Quiz set – evaluates teams' submissions for leaderboard
    - Test set – used by Netflix to determine winner

Additional Evaluations in e-commerce

- Total sales numbers
- Promotion of certain items
- …
- Click-through-rates
- Interactivity on platform
- …
- Customer return rates
- Customer satisfaction and loyalty

An imperfect world

- Offline evaluation is the cheapest variant
  - Still, gives us valuable insights
  - and lets us compare our results (in theory)
- Alternative and complementary measures:
  - Diversity, Coverage, Novelty, Familiarity, Serendipity, Popularity, Concentration effects (Long tail)

References

- Recommender Systems: An Introduction
  - http://www.recommenderbook.net/
  - Slides from Introduction, collaborative recommendations and evaluation of RS chapters.

- Other reading materials:
  - http://www.inf.unibz.it/~ricci/ISR/papers/cf-adaptive-web.pdf
  - http://www.inf.unibz.it/~ricci/ISR/papers/handbook-neighbor.pdf

Thank You!