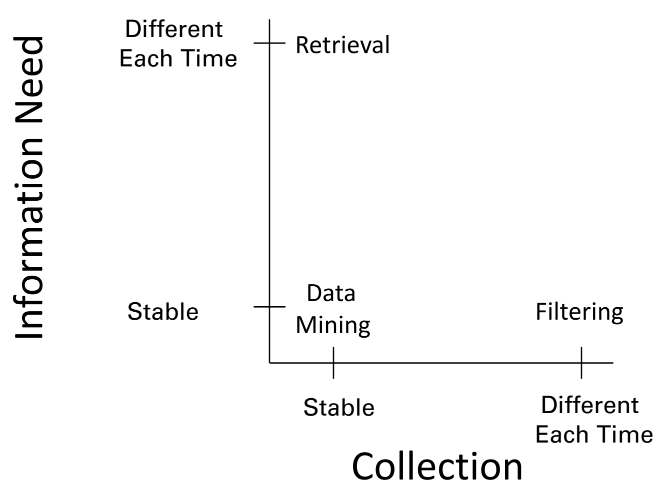
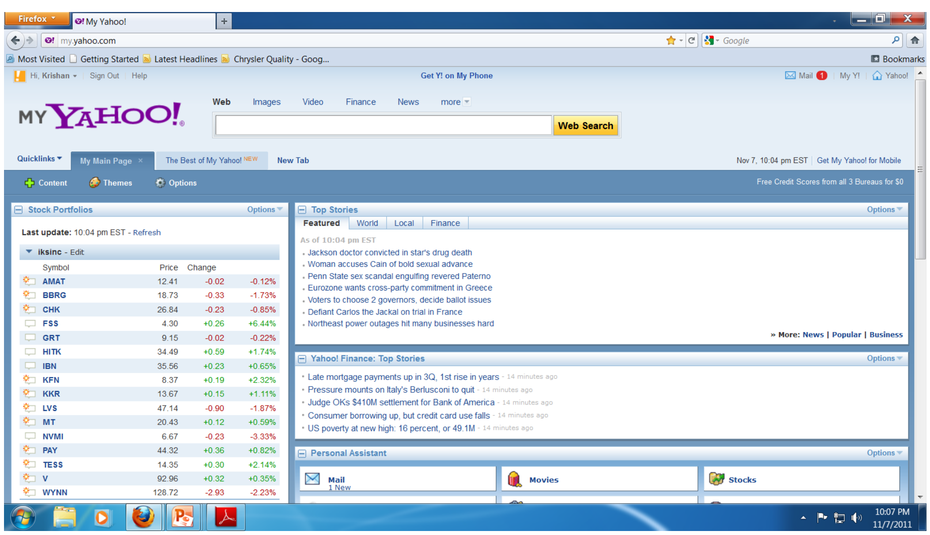
Recommender Systems & Collaborative Filtering

Information Access Problems



Information Retrieval Vs. Information Filtering

- An Information retrieval systems responds by presenting retrieved documents or links to documents in response to user queries that change according to user needs. The collection of documents may or may not be static
- An Information filtering system retrieves documents in response to a fixed user query. The collection in an information filtering system is dynamic
  - Example: Your personalized Yahoo! Page (My Yahoo!)



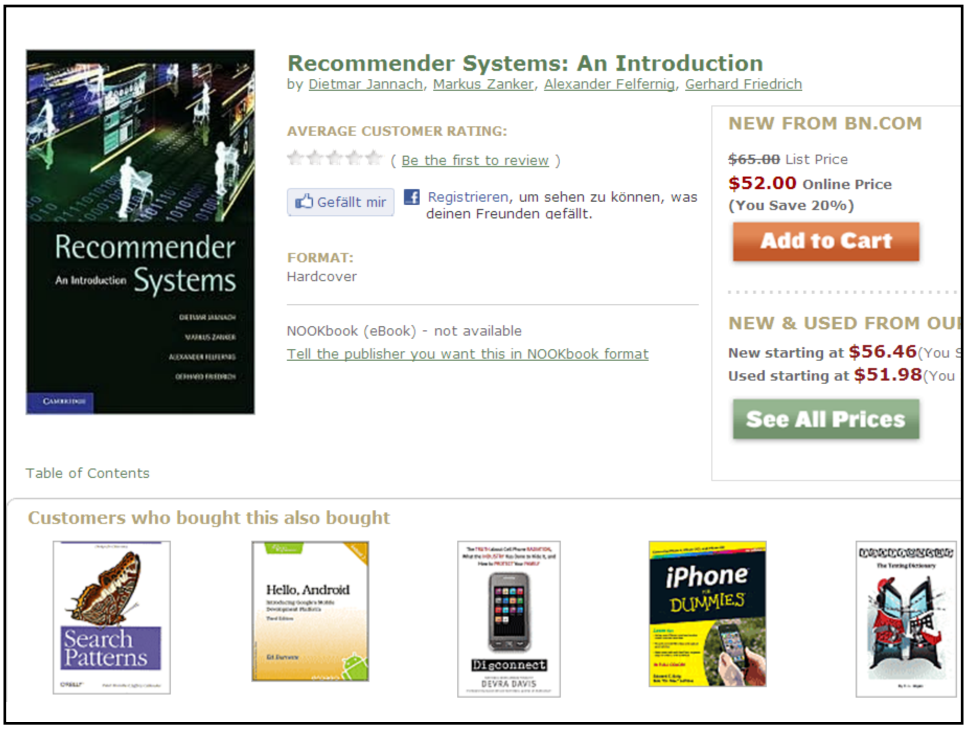
Information filtering is content based.

What is a Recommender System?

- When the delivered information comes in the form of suggestions an information filtering system is called a recommender system.

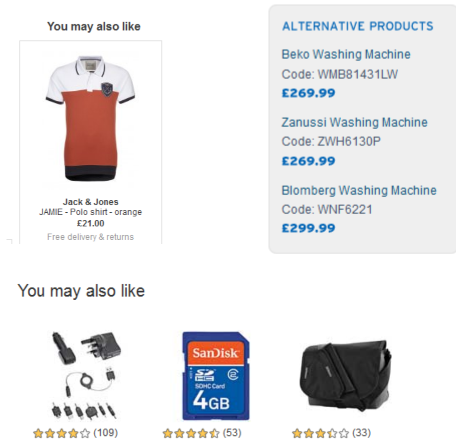
A recommendation system uses collaborative/content filtering to personalize product predictions for users.

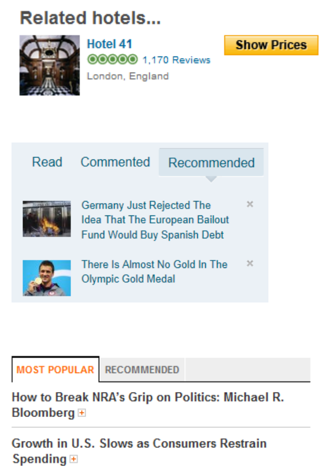
On September 21, 2009 "BellKor's Pragmatic Chaos" team won the $1M Grand Prize offered by Netflix to improve prediction accuracy for enjoying a movie based on collaborative filtering



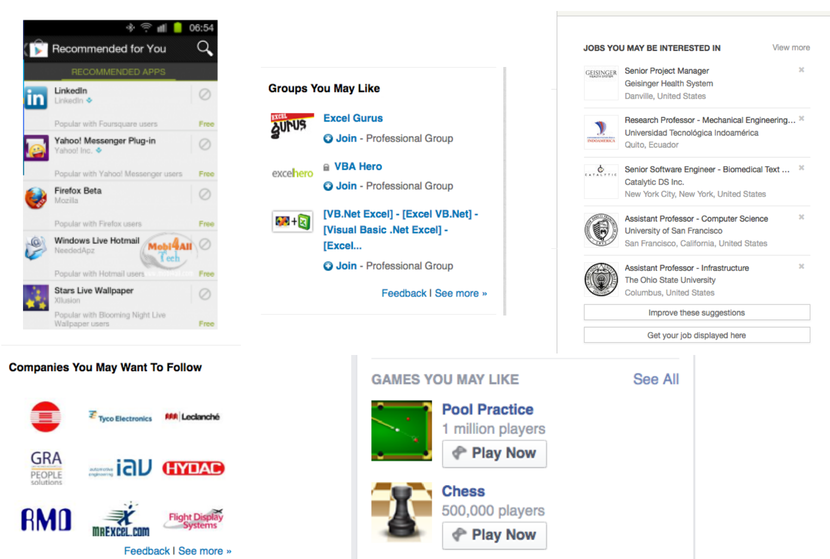
Recommender Systems

- Application areas





In the Social Web



Recommender Systems Success Stories

- 35% of the purchases on Amazon are the result of their recommender system
- Recommendations are responsible for 70% of the time people spend watching videos on YouTube
- 75% of what people are watching on Netflix comes from recommendations
- 38% more click-thru due to recommendations estimated by Google

Recommendation System Approaches

- Collaborative filtering
  - Locate users with similar preferences to predict how well the item under consideration will be received
- Content based
  - Uses the features of the item under consideration to predict how well it will be received
- Demographic based
  - Incorporates users demographics into consideration

Collaborative Filtering

- A way of making suggestions for information/products based on community input
- Everyday examples of collaborative filtering
  - Best sellers list
  - Unmarked but well-used paths thru the woods
  - Top ten downloads from a freeware site
  - Popular movies at a rental site/store

User-based 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

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Also known as memory based filtering

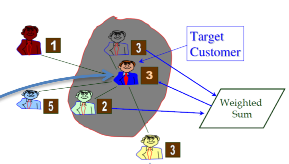
User-based Collaborative Filtering (2)

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

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

User-based Collaborative Filtering Algorithm

- $v_{i,j}$= vote of user *i* on item *j*
- I_i = items for which user *i* has voted
- Mean vote for user *i* is



$$\overline{v}_i = \frac{1}{|I_i|}\sum_{j \in I_i} v_{i,j}$$

- Predicted vote for "active user" *a* is weighted sum

$$p_{a,j} = \overline{v}_a + \kappa \sum_{i=1}^{n} \underbrace{w(a,i)}_{\text{weights of } n \text{ similar users}} (v_{i,j} - \overline{v}_i)$$

normalizer

Selecting Weights and Neighbors

- K-nearest neighbor

$$w(a,i) = \begin{cases} 1 & \text{if } i \in \text{neighbors}(a) \\ 0 & \text{else} \end{cases}$$

- Pearson correlation coefficient (Resnick ’94, Grouplens):

$$w(a,i) = \frac{\sum_j (v_{a,j} - \overline{v}_a)(v_{i,j} - \overline{v}_i)}{\sqrt{\sum_j (v_{a,j} - \overline{v}_a)^2 \sum_j (v_{i,j} - \overline{v}_i)^2}}$$

- Cosine distance (from IR)

$$w(a,i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}}$$

Selecting Weights

- Cosine with “inverse user frequency” $f_i = log(n/n_j)$, where n is number of users, n_j is number of users voting for item j

$$w(a,i) = \frac{\sum_j f_j \sum_j f_j v_{a,j} v_{i,j} - (\sum_j f_j v_{a,j})(\sum_j f_j v_{i,j}))}{\sqrt{UV}}$$

where

$$U = \sum_j f_j (\sum_j f_j v_{a,j}^2 - (\sum_j f_j v_{a,j})^2)$$

$$V = \sum_i f_j (\sum_i f_j v_{i,j}^2 - (\sum_i f_j v_{i,j})^2)$$

Measuring user similarity example

- **Pearson correlation**

$$w(a,i) = \frac{\sum_j (v_{a,j} - \overline{v}_a)(v_{i,j} - \overline{v}_i)}{\sqrt{\sum_j (v_{a,j} - \overline{v}_a)^2 \sum_j (v_{i,j} - \overline{v}_i)^2}}$$

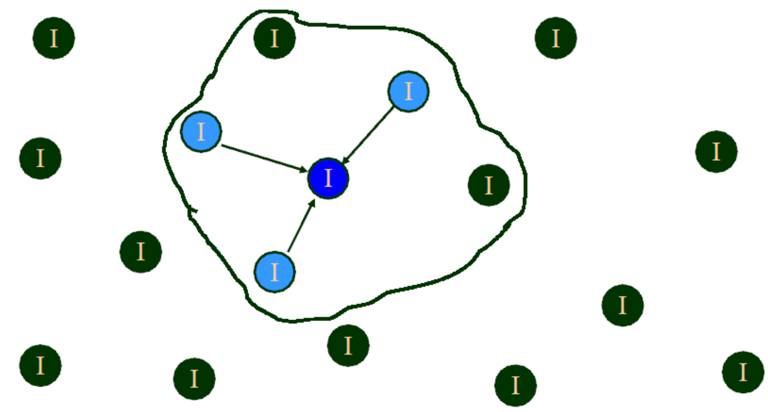
	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = 0,85
sim = 0,70
sim = -0,79

Improving the metrics / prediction function

- Not all neighbor 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
- 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
- Case amplification
  - Intuition: Give more weight to "very similar" neighbors, i.e., where the similarity value is close to 1.
- Neighborhood selection
  - Use similarity threshold or fixed number of neighbors

Item Based Collaborative Filtering Algorithm



Looks for items similar to those that a user has previously liked/bought

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 Item5
  - Take Alice's ratings for these items to predict the rating for Item5

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Also known as content-based filtering

Pre-processing for item-based filtering

- Pre-processing approach by Amazon.com (in 2003)
  - Calculate all pair-wise 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 pair-wise similarities to be memorized (N = number of items) in theory
  - In practice, this is significantly lower (items with no co-ratings)
  - Further reductions possible
    - Minimum threshold for co-ratings (items, which are rated at least by *n* users)
    - Limit the size of the neighborhood (might affect recommendation accuracy)

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
  - 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 ***i*** for the neighbor
    - Use this predicted rating instead of the rating of a more distant direct neighbor

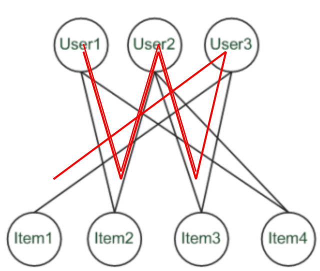
	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	?
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = 0,85

Predict rating for User1

Graph-based methods

- "Spreading activation" (sketch)
  - Idea: Use paths of lengths > 3 to recommend items
  - Length 3: Recommend Item3 to User1
  - Length 5: Item1 also recommendable

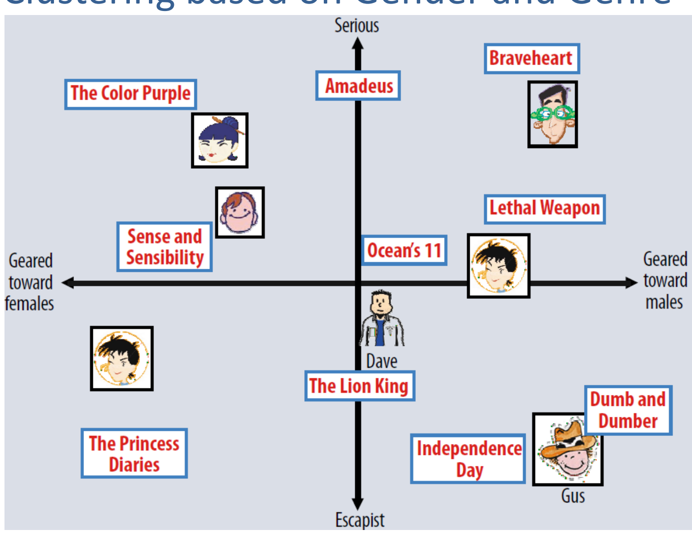


Demographic Filtering

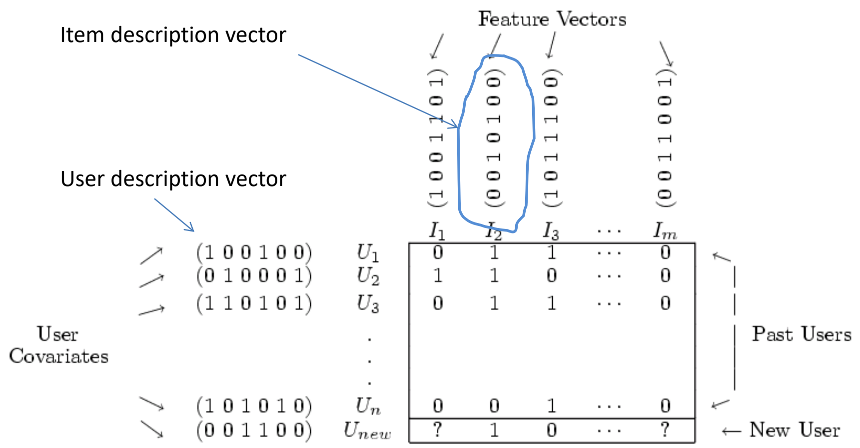
- Perform clustering to divide the customer base into many segments and treat the task as a classification problem.
- The algorithm's goal is to assign the user to the segment containing the most similar customers.
- It then uses the purchases and ratings of the customers in the segment to generate recommendations.

Demographic Filtering Example

- Clustering based on Gender and Genre



Collaborative + Content Filtering



Collaborative + Content Filtering

		Airplane	Matrix	Room with a View	...	Hidalgo
		comedy	action	romance	...	action
Joe	*27,M,70k*	9	7	2		7
Carol	*53,F,20k*	8		9		
...						
Kumar	*25,M,22k*	9	3			6
U_a	*48,M,81k*	4	7	?	?	?

Collaborative + Content Filtering As Classification (Basu, Hirsh, Cohen, AAAI98)

Classification task: map **(user,movie)** pair into **{likes,dislikes}**

Training data: known likes/dislikes

Test data: active users

Features: **any** properties of user/movie pair

		Airplane	Matrix	Room with a View	...	Hidalgo
		comedy	action	romance	...	action
Joe	*27,M,70k*	1	1	0		1
Carol	*53,F,20k*	1		1		0
...						
Kumar	*25,M,22k*	1	0	0		1
U_a	*48,M,81k*	0	1	?	?	?

IEEE Computer, August 2009

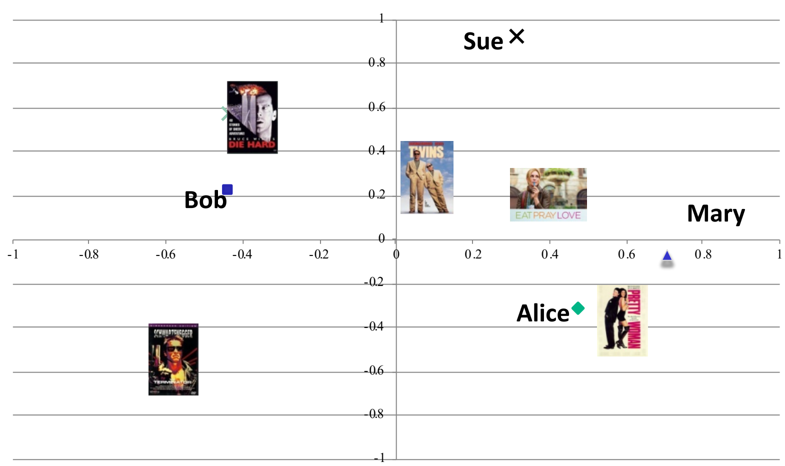
MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS

Yehuda Koren, *Yahoo Research*
Robert Bell and Chris Volinsky, *AT&T Labs—Research*

Basic Idea

- Each item (movie in this case) is associated with a vector, $\mathbf{q}_i$, of m components.
- Each user is associated with a profile vector, $\mathbf{u}_j$.
- The dot product $\mathbf{q}_i^t u_j$ captures the interaction between an item-user pair
- We can thus form an item-user matrix similar to term-document matrix and apply SVD/LSI
- Issue: The matrix has many blanks. Previous approaches tried predicting the missing values followed by SVD
- Netflix Paper Approach: Optimize the prediction error for known ratings and in the process fill the missing values

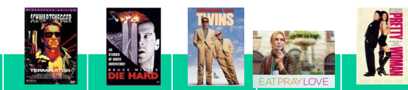
A picture says …



Matrix factorization

- SVD: $M_k = U_k \times \Sigma_k \times V_k^T$

U_k	Dim1	Dim2
Alice	0.47	-0.30
Bob	-0.44	0.23
Mary	0.70	-0.06
Sue	0.31	0.93



V_k^T					
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

Σ_k	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

- Prediction: $\hat{r}_{ui} = \bar{r}_u + U_k(Alice) \times \Sigma_k \times V_k^T(EPL)$

  = 3 + 0.84 = 3.84

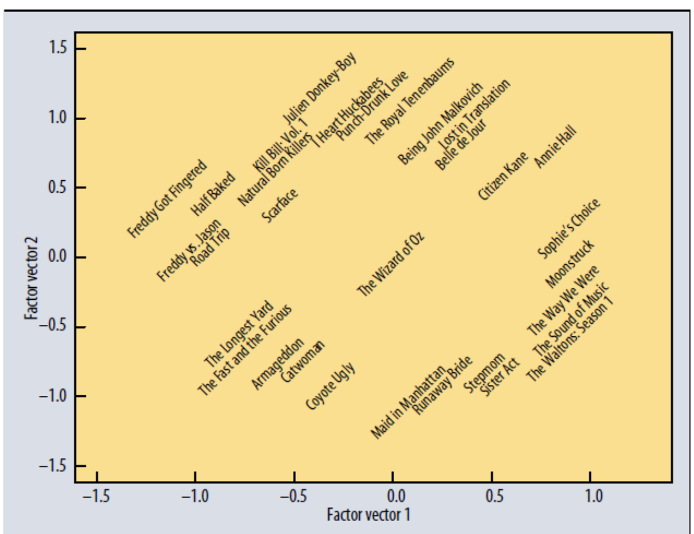
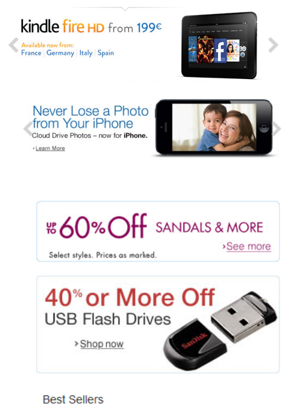


Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films.

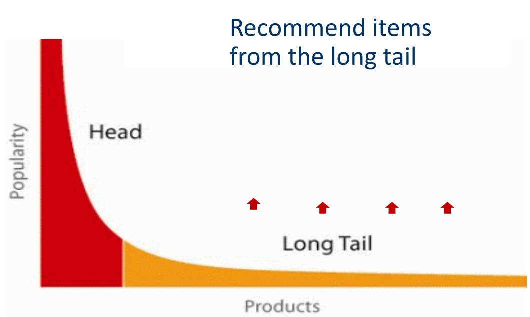
How to evaluate recommendations?

What are the measures in practice?

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



When does a RS do its job well?



- "Recommend widely unknown items that users might actually like!"
- 20% of items accumulate 74% of all positive ratings

Evaluation in information retrieval (IR)

- Recommendation is viewed as information retrieval task:
  - Retrieve (recommend) all items which are predicted to be "good" or "relevant".
- Common protocol :
  - Hide some items with known ground truth
  - Rank items or predict ratings -> Count -> Cross-validate
- Ground truth established by human domain experts

		Reality	
		Actually Good	Actually Bad
Prediction	Rated Good	True Positive (tp)	False Positive (fp)
	Rated Bad	False Negative (fn)	True Negative (tn)

Accuracy measures

- Datasets with items rated by users
  - MovieLens datasets 100K-10M ratings
  - Netflix 100M ratings
- Historic user ratings constitute ground truth
- Metrics measure error rate
  - 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

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

Challenges in Recommendation algorithms

- A large retailer might have huge amounts of data, tens of millions of customers and millions of distinct catalog items.
- Many applications require the results set to be returned in real-time, in no more than half a second, while still producing high-quality recommendations.
- New customers typically have extremely limited information, based on only a few purchases or product ratings.
- Older customers can have a glut of information, based on thousands of purchases and ratings.
- Customer data is volatile: Each interaction provides valuable customer data, and the algorithm must respond immediately to new information.