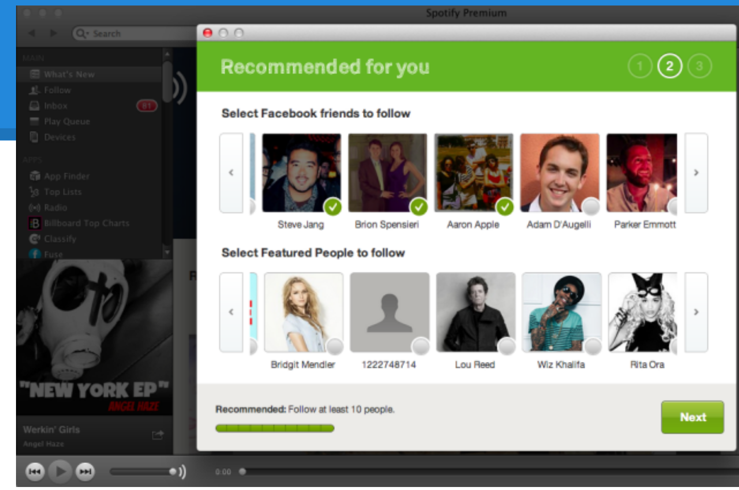
Algorithms in Recommendation Systems

Presented by:
Dumitrel Loghin, Anuja Meetoo Appavoo, Suhendry Effendy, Paramasiven Appavoo, Lu Bingxin, Li Jing, Suman Sourav

Familiar interfaces …



What are Recommendation Systems?

- systems (algorithms) trying to predict user preferences for new items

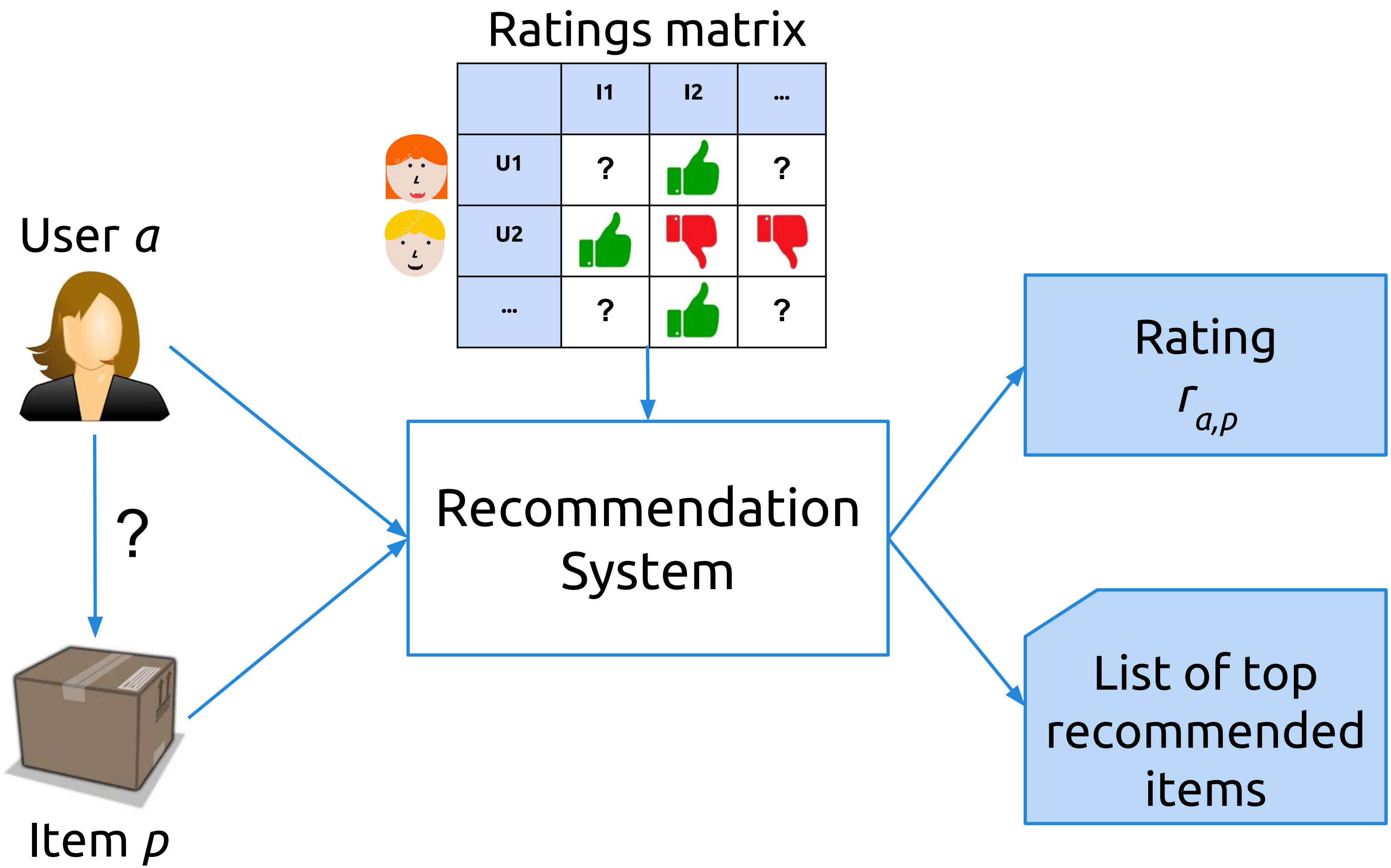
- all modern web apps have a recommender system
  - books and items (Amazon)
  - music (Spotify)
  - movies (IMDB)
  - friends (Facebook)
  - ...

Why using Recommendation Systems?

Approaches

- Collaborative filtering
  - user-based
  - item-based
  - major challenges
- Content-based
- Knowledge-based

Collaborative filtering



Collaborative filtering

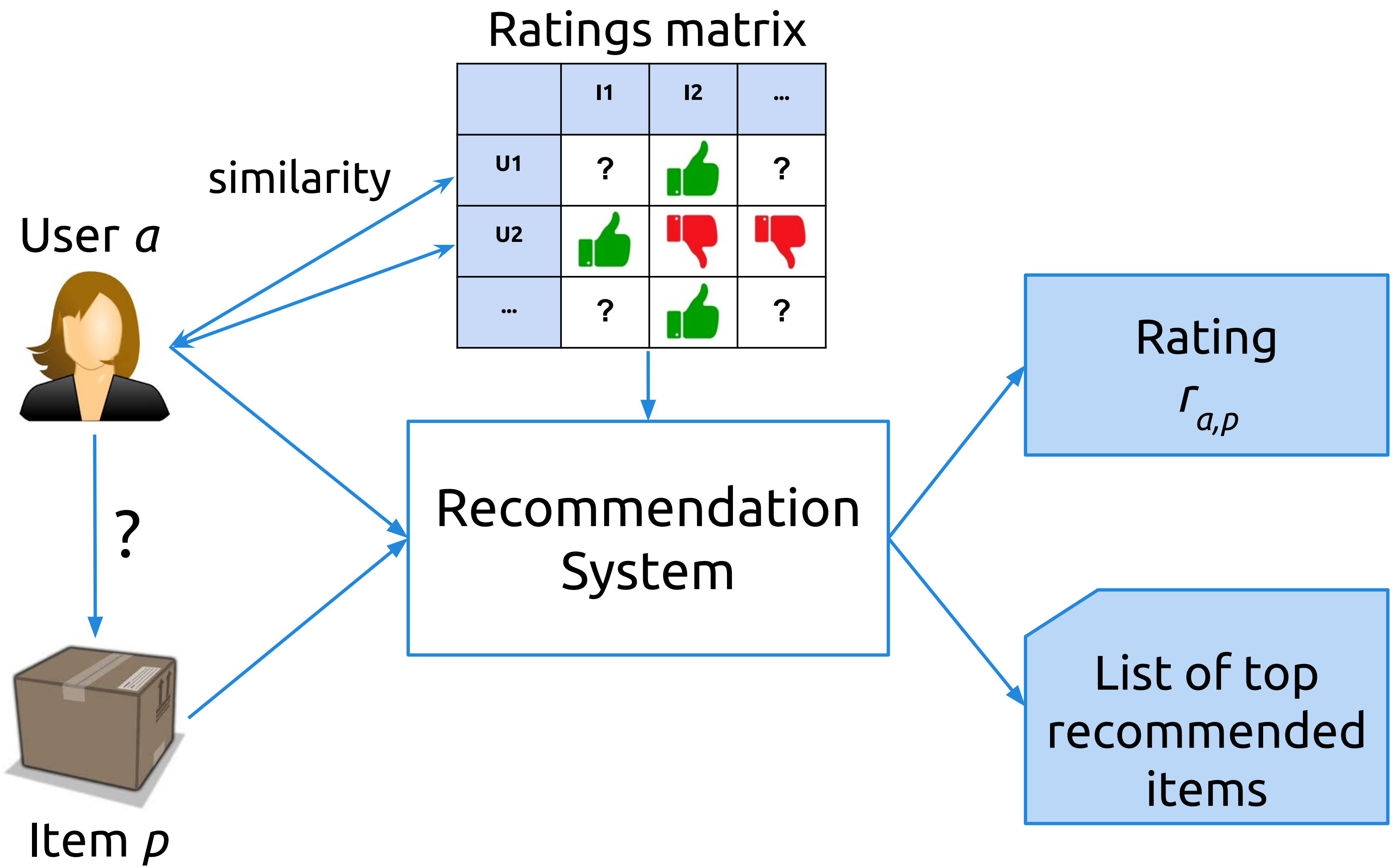
- Assumptions
  - users rate items explicitly or implicitly
  - user's taste preserves over time
  - use other users ratings (wisdom of the crowd)

- Approach
  - user-item matrix
  - predict the rating for a particular item or compute a list of recommended items
    - based on other users ratings
    - based on similar items ratings

User-based Collaborative Filtering

Dumitrel

User-based collaborative filtering



Similarity and prediction equations

- Active user *a* and item *p*, $r_{a,p}$ is unknown
- First, compute similarities with other users

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

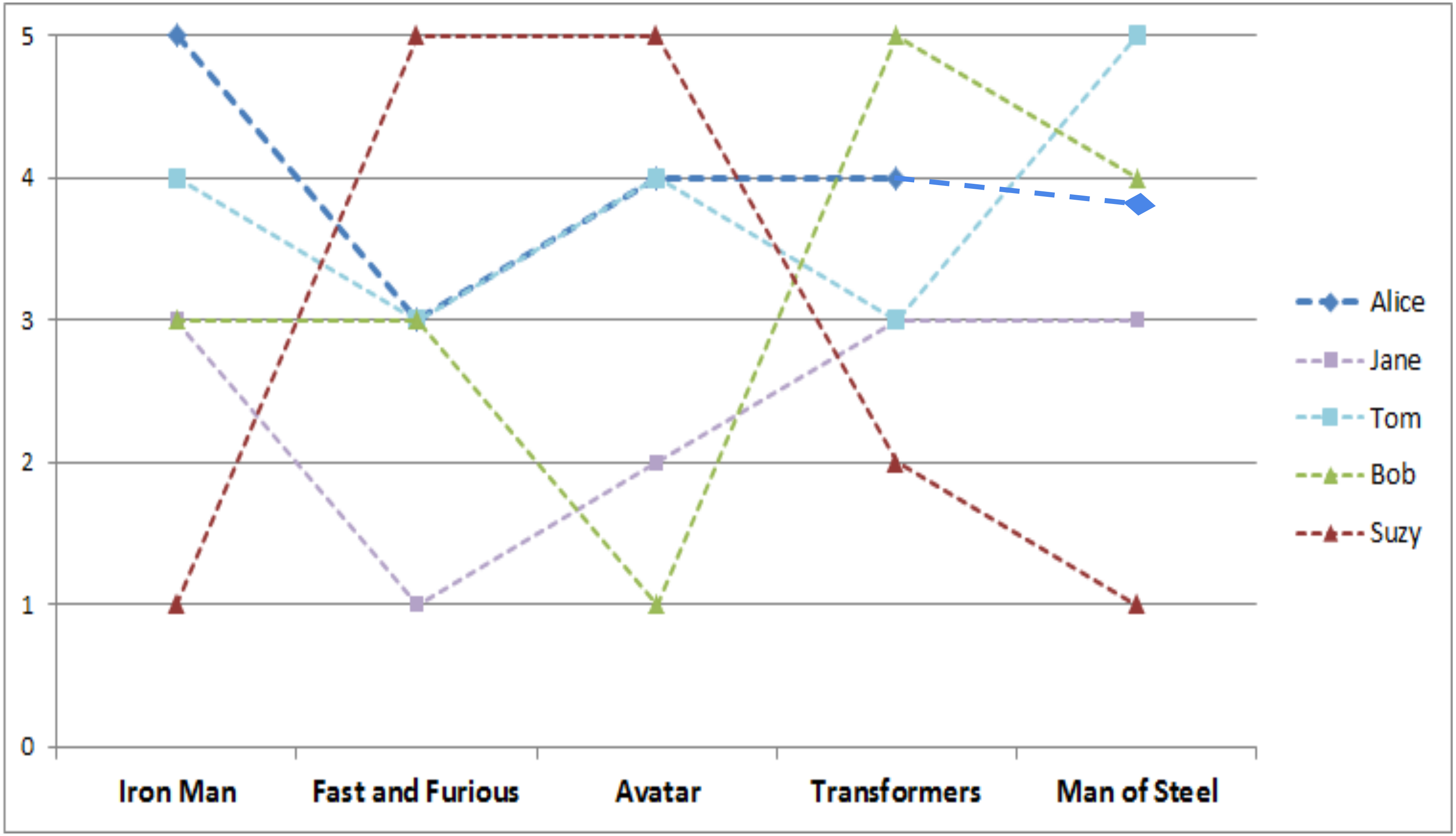
- Then predict $r_{a,p}$

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

Example

similarity		Iron Man	Fast and Furious	Avatar	Transformers	Man of Steel
	Alice	5	3	4	4	?
0.85	Jane	3	1	2	3	3
0.70	Tom	4	3	4	3	5
0	Bob	3	3	1	5	4
-0.79	Suzy	1	5	5	2	1

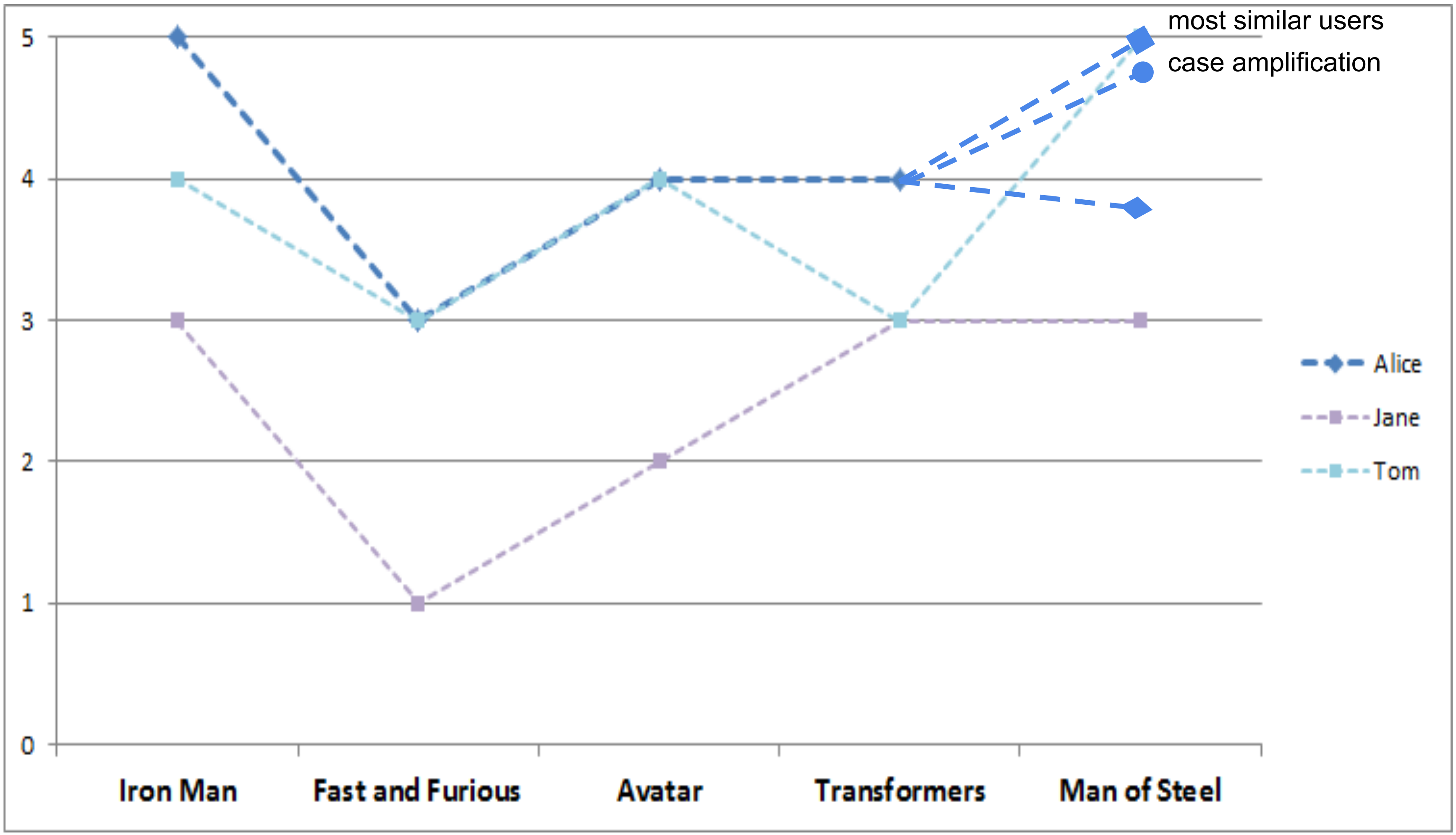
User similarities



Possible improvements

- Better similarity and prediction functions
  - give more weight to items having diverse ratings
  - give more weight to similar users - case amplification
- Better neighborhood selection
  - select only the most similar users
  - select users with whom the user has more common rated items

Using only similar users



Algorithm

	Steps
UserBasedCF(User a, Item p)	
compute average rating for a	m
for each b in all other users	n
if b purchased p	
compute average rating for b	m
compute *sim(a,b)*	m
[select the neighborhood]	O(n)
compute rating $r_{a,p}$	m
	....
	O(nm)

Collaborative filtering challenges

- Scalability - huge rating matrix
  - 10^8 users and growing
  - 10^7 items and growing fast
- Sparsity - many undefined ratings
- Cold start - new users, new items
- Conspiracy - users agreement or shilling attacks using bots
- Privacy - user profiles

Item-based Collaborative Filtering

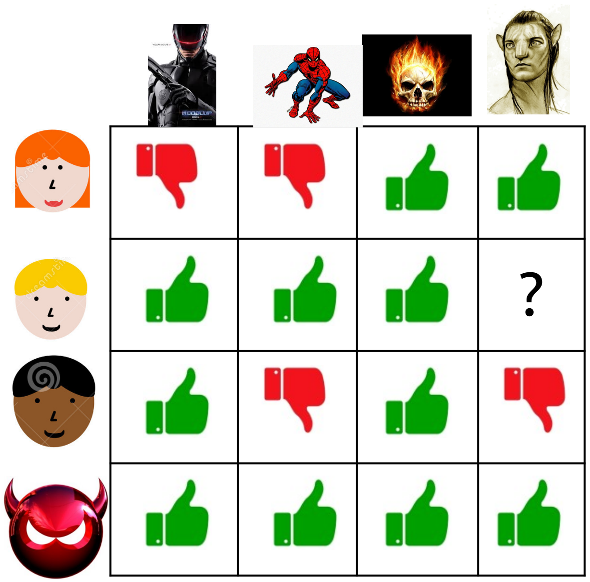
Anuja

Shilling Attack

- User-based collaborative filtering is vulnerable to attack
  - Rely on user specified judgements (anyone)
  - Fake user profile to manipulate ratings
    - Push attack: Increase rating of one's items
    - Nuke attack: Lower rating of competitors' items

- Biased recommendation
  - Decrease user satisfaction

- Real case: Sony Pictures admitted it used fake quotes from non-existent movie critics to promote a number of newly released films (June 2001)



Scalability

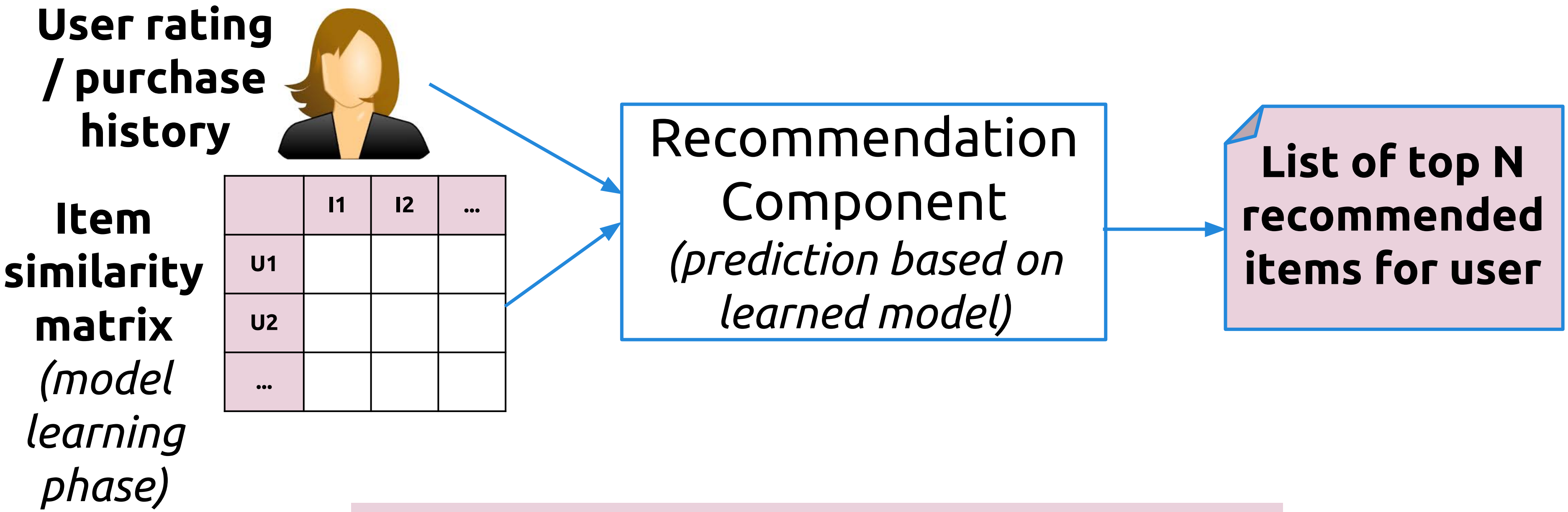
- E-commerce recommendation systems often operate in a challenging environment
  - Millions of users and catalog items
  - High quality recommendations needed in real-time
- User-based collaborative filtering
  - Need to scan vast no. of neighbours
  - **Real-time prediction infeasible!**
  - Does not scale for most real-world scenarios :(

How does Amazon handle all its users and catalog items???

amazon.com

Item-based collaborative filtering

- Use similarity between items to predict user ratings
    - Item similarities are considered to be more stable than user similarities (Sarwar et al. 2001)



Underlying principle: We tend to buy products similar to what we like.

Example

	Iron Man	Fast and Furious	Avatar	Transformers	Man of Steel
Alice	5	3	4	4	
Jane	3	1	2	3	3
Tom	4	3	4	3	5
Bob	3	3	1	5	4
Suzy	1	5	5	2	1

- Look for movies (items) similar to Man of Steel
- Use Alice’s ratings for these movies to predict her rating for Man of Steel

Example

	Iron Man	Fast and Furious	Avatar	Transformers	Man of Steel
Alice	5	3	4	4	?
Jane	3	1	2	3	3
Tom	4	3	4	3	5
Bob	3	3	1	5	4
Suzy	1	5	5	2	1

- **Look for movies (items) similar to Man of Steel**
- Use Alice's ratings for these movies to predict her rating for Man of Steel

Example

	Iron Man	Fast and Furious	Avatar	Transformers	Man of Steel
Alice	5	3	4	4	?
Jane	3	1	2	3	3
Tom	4	3	4	3	5
Bob	3	3	1	5	4
Suzy	1	5	5	2	1

- **Look for movies (items) similar to Man of Steel**
- Use Alice's ratings for these movies to predict her rating for Man of Steel

Example

	Iron Man	**Fast and Furious**	Avatar	Transformers	Man of Steel
Alice	5	3	4	4	?
Jane	3	1	2	3	3
Tom	4	3	4	3	5
Bob	3	3	1	5	4
Suzy	1	5	5	2	1

- **Look for movies (items) similar to Man of Steel**
- Use Alice's ratings for these movies to predict her rating for Man of Steel

Example

	Iron Man	**Fast and Furious**	Avatar	Transformers	Man of Steel
Alice	5	3	4	4	?
Jane	3	1	2	3	3
Tom	4	3	4	3	5
Bob	3	3	1	5	4
Suzy	1	5	5	2	1

- **Look for movies (items) similar to Man of Steel**
- Use Alice's ratings for these movies to predict her rating for Man of Steel

Example

	Iron Man	Fast and Furious	**Avatar**	Transformers	Man of Steel
Alice	5	3	4	4	?
Jane	3	1	2	3	3
Tom	4	3	4	3	5
Bob	3	3	1	5	4
Suzy	1	5	5	2	1

- **Look for movies (items) similar to Man of Steel**
- Use Alice's ratings for these movies to predict her rating for Man of Steel

Example

	Iron Man	Fast and Furious	**Avatar**	Transformers	Man of Steel
Alice	5	3	4	4	?
Jane	3	1	2	3	3
Tom	4	3	4	3	5
Bob	3	3	1	5	4
Suzy	1	5	5	2	1

- **Look for movies (items) similar to Man of Steel**
- Use Alice's ratings for these movies to predict her rating for Man of Steel

Example

	Iron Man	Fast and Furious	Avatar	**Transformers**	Man of Steel
Alice	5	3	4	4	?
Jane	3	1	2	3	3
Tom	4	3	4	3	5
Bob	3	3	1	5	4
Suzy	1	5	5	2	1

- **Look for movies (items) similar to Man of Steel**
- Use Alice's ratings for these movies to predict her rating for Man of Steel

Example

	Iron Man	Fast and Furious	Avatar	**Transformers**	Man of Steel
Alice	5	3	4	4	?
Jane	3	1	2	3	3
Tom	4	3	4	3	5
Bob	3	3	1	5	4
Suzy	1	5	5	2	1

- **Look for movies (items) similar to Man of Steel**
- Use Alice's ratings for these movies to predict her rating for Man of Steel

Example

	Iron Man	Fast and Furious	Avatar	**Transformers**	Man of Steel
Alice	$\boxed{5}$	3	4	$\boxed{4}$	?
Jane	3	1	2	3	3
Tom	4	3	4	3	5
Bob	3	3	1	5	4
Suzy	1	5	5	2	1

- Look for movies (items) similar to Man of Steel
- **Use Alice's ratings for these movies to predict her rating for Man of Steel**

You must be wondering...

How can similar items be identified?

How can user rating be predicted based on predictions of similar items?

Identifying similar items

- Cosine similarity measure
- Ratings are seen as vector in n-dimensional space
- Similarity calculated based on angle between 2 vector

- Similarity between 2 items a and b

Similarity values are between 0 and 1, where values near to 1 indicate strong similarity

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

Note:

  - '·' is dot product
  - Euclidean length, defined as $\|\vec{x}\| = \sqrt{x_1^2 + \cdots + x_n^2}$

Differences in average rating behavior of users not considered!!!
(Some users may generally give high ratings while others may give lower ratings as a preference)

Adjusted cosine measure

- Takes average user ratings into account
- Subtracts user average from ratings

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

Note: U refers to set of users having rated items a and b

- Values range from –1 to +1, as in Pearson measure

Similarity for items with only one common user is 1
Only items with one common user end up being most similar :(
Solution: Need to have a minimum number of users in common for 2 items to be considered for similarity

Adjusted cosine measure - Example

	Iron Man	Fast and Furious	Avatar	Transformers	Man of Steel
Alice	1.00	-1.00	0.00	0.00	?
Jane	0.60	-1.40	-0.40	0.60	0.60
Tom	0.20	-0.80	0.20	-0.80	1.20
Bob	-0.20	-0.20	-2.20	2.80	0.80
Suzy	-1.80	2.20	2.20	-0.80	-1.80

Adjusted cosine similarity value for Man of Steel and Iron Man:

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

Adjusted cosine measure - Example

	Iron Man	Fast and Furious	Avatar	Transformers	Man of Steel
Alice	5	3	4	4	?
Jane	3	1	2	3	3
Tom	4	3	4	3	5
Bob	3	3	1	5	4
Suzy	1	5	5	2	1
Similarity	**0.80**	**-0.90**	**-0.76**	**0.42**	

Predicting user rating

- Calculate weighted sum of Alice's ratings for movies (items) similar to Man of Steel

$$pred(u,p) = \frac{\sum_{i \in L(u)} sim(i,p) \cdot r_{u,i}}{\sum_{i \in L(u)} sim(i,p)}$$

L(u) = list of similar items to item p rated by user u

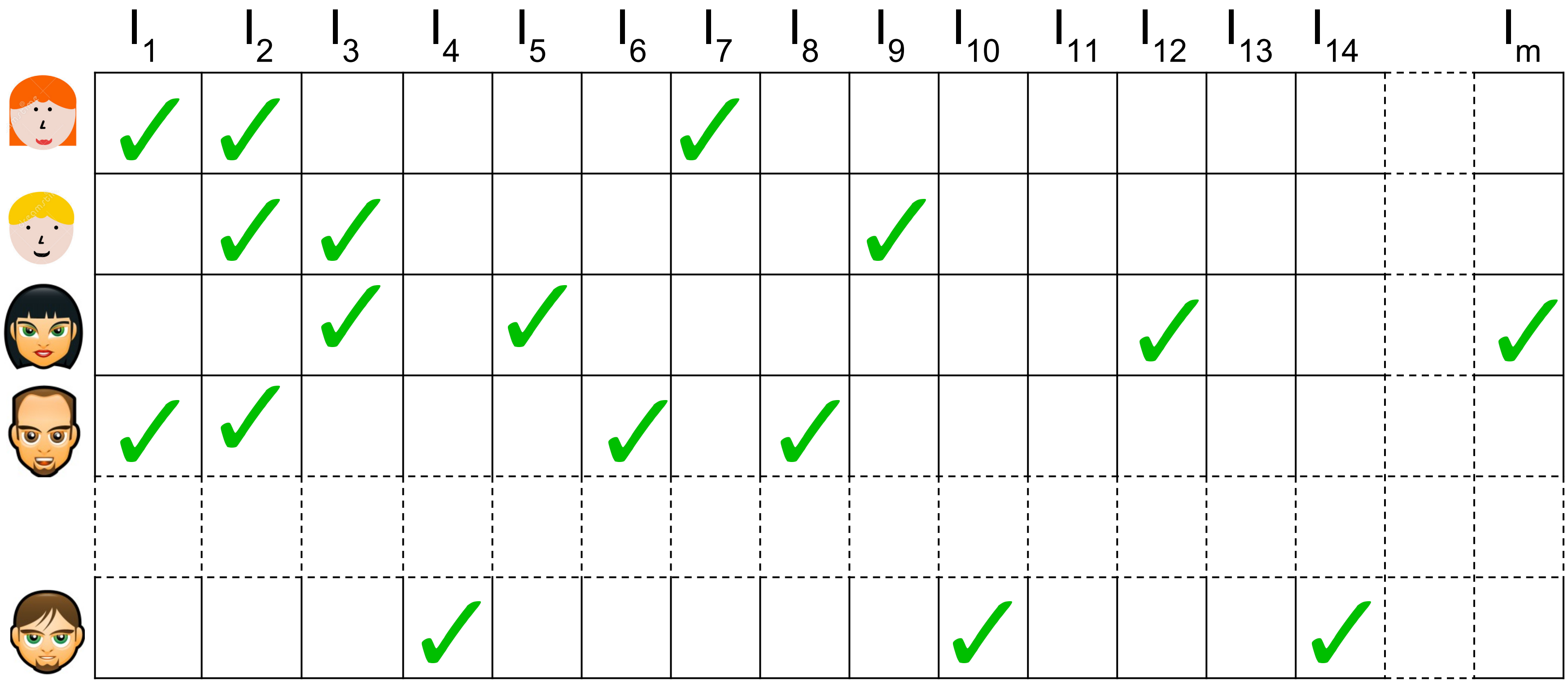
- Number of similar items considered for prediction limited to a specific size
  - similar idea to user-based collaborative filtering

Predicting user rating - Example

	Iron Man	Transformers	Avatar	Fast and Furious	Man of Steel
Alice	5	3	4	4	?
Jane	3	1	2	3	3
Tom	4	3	4	3	5
Bob	3	3	1	5	4
Suzy	1	5	5	2	1
Similarity	**0.80**	**-0.90**	**-0.76**	**0.42**	

- pred(Alice, Man of Steel)
  = ((0.80*5) + (0.42 * 4)) / (0.80 + 0.42) = 4.7

In practice... sparse matrix



- Customers have very few purchases / rate very few items

Item-based collaborative filtering algo

	Worst	In practice
Compute item similarity matrix	**$O(m^2n)$**	**$O(mn)$**
For each item in product catalog, I_p	*m*	*m*
For each customer C who purchased I_p	*n*	<< *n*
For each item I_q purchased by customer C	*m - 1*	<< *m*
Record that a customer purchased I_p and I_q	*c*	*c*
For each item I_q	*m - 1*	<< *m*
Compute the similarity between I_p and I_q	*n*	<< *n*
Predict the user rating of product	**O(m)**	**O(m)**
Generate the list of top recommended items for user	**O(m)**	**O(m)**

(n: no of customers, m: no of catalog items)

Space requirements: O(mn)

Item-based collaborative filtering

- Scales independently of no of users or items
  - Depend only on no of items rated by active user

- Less affected by shilling attack (Lam and Riedl 2004)
  - Predicted rating for item determined by comparing its item vector with those of other items
  - Attacker has no control over ratings given by other users to any item

	I_1	I_2	I_3	I_4	I_5	I_6	**Correlation with Alice**	
Alice *(active user)*	5	2	3	3		?		
	2		4		4	1	-1.00	
	3	1	3		1	2	0.76	
	4	2	3	1		1	0.72	
	3	3	2	1	3	1	0.21	
		3		1	2		-1.00	
	4	3		3	3	2	0.94	*most similar user without attack*
		5		1	5	1	-1.00	
	5		3		2	5	1.00	*most similar user with attack*
	5	1	4		2	5	0.89	
	5	2	2	2		5	0.93	
Correlation with I_6	0.85	-0.55	0.00	0.48	-0.59			
	popular item							

Collaborative Filtering
Major Challenges

Suhendry Effendy & Paramasiven

Scalability Problem

User-based CF : $O(n^2m)$ at worst, or $O(n^2)$ in practice.

Item-based CF : $O(nm^2)$ at worst, or $O(nm)$ in practice.

How to deal with millions of users and items?

Scalability Problem

User-based CF : $O(n^2m)$ at worst, or $O(n^2)$ in practice.

Item-based CF : $O(nm^2)$ at worst, or $O(nm)$ in practice.

How to deal with millions of users and items?

Several approaches:

Clustering CF

Bayesian CF

Regression-Based CF

MDP-Based CF, etc.

Clustering CF

Users or items are grouped by their similarity.

	I_1	I_2	I_3	I_4	I_5	I_6
U_1	x		x	x		
U_2	x		x	x	x	
U_3	x	x		x	x	
U_4			x		x	x
U_5			x	x	x	x
U_6			x		x	

		I_1	I_2	I_3	I_4	I_5	I_6
	U_1	x		x	x		
cluster 1	U_2	x		x	x	x	
	U_3	x	x		x	x	
	U_4			x		x	x
cluster 2	U_5			x	x	x	x
	U_6			x		x	

user clustering

	cluster 1			cluster 2		
	I_1	I_2	I_3	I_4	I_5	I_6
U_1	x		x	x		
U_2	x		x	x	x	
U_3	x	x		x	x	
U_4			x		x	x
U_5			x	x	x	x
U_6			x		x	

item clustering

Clustering CF

How to make use the clustering?

- only consider users/items in the same cluster.
- smaller size = faster running time

		I_1	I_2	I_3	I_4	I_5	I_6
cluster 1	U_1	x		x	x		
	U_2	x		x	x	x	
	U_3	x	x		x	x	
cluster 2	U_4			x		x	x
	U_5			x	x	x	x
	U_6			x		x	

	cluster 1			cluster 2		
	I_1	I_2	I_3	I_4	I_5	I_6
U_1	x		x	x		
U_2	x		x	x	x	
U_3	x	x		x	x	
U_4			x		x	x
U_5			x	x	x	x
U_6			x		x	

Clustering CF

Item Clustering

- ➔ based on item type (e.g., books, gadgets, etc.)
- ➔ based on item similarity.

User Clustering

- ➔ based on user’s similarity.

similarity

- ≠ similar rates or preferences
- = rate similar set of items

Clustering CF

Item Clustering

- ➔ based on item type (e.g., books, gadgets, etc.)
- ➔ based on item similarity.

User Clustering

- ➔ based on user’s similarity.

Jaccard Similarity

$$\frac{|A \cap B|}{|A \cup B|}$$

similarity

≠ similar rates or preferences

= rate similar set of items

Cosine Similarity

$$\frac{A \cdot B}{\|A\| \|B\|}$$

Clustering Algorithm for CF

Modularity Maximization (Newman)

- ➔ NP-Hard Problem (Brandes et al.)
- ➔ usually used in community detection problem.
- ➔ constant approximation algorithm (Dinh and Thai)
- ➔ proposed by Pham et al. for clustering CF

RecTree (Chee et al.)

- ➔ recursive CF clustering
- ➔ K-Means

Modularity Maximization

Modularity is the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random.

$$Q = \sum_{i=1}^{q} (e_{ii} - r_i)$$

e_{ii} = % of edges in cluster i

r_i = probability of random edge belong to cluster i

Modularity Maximization

Modularity is the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random.

probability random edge belong to cluster i

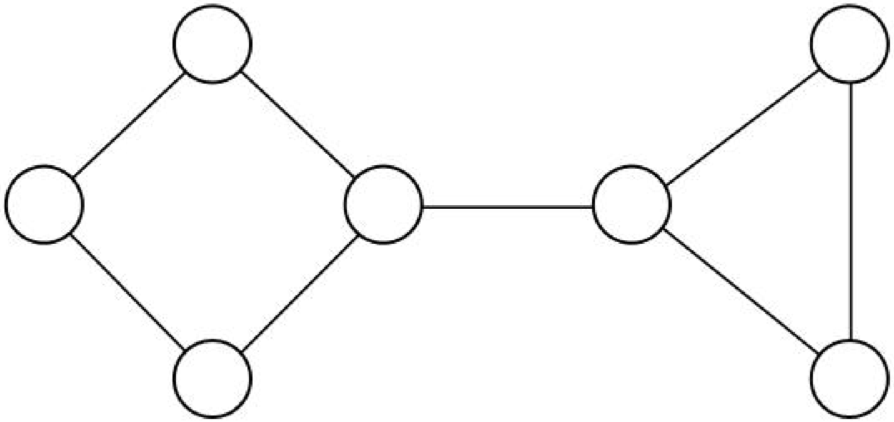
$$Q = \sum_{i=1}^{q} (e_{ii} - a_i{}^2)$$

random rewire proportional to node's degree

e_{ii} = % of edges in cluster i

a_i = % of degree of nodes in cluster i

Modularity Maximization

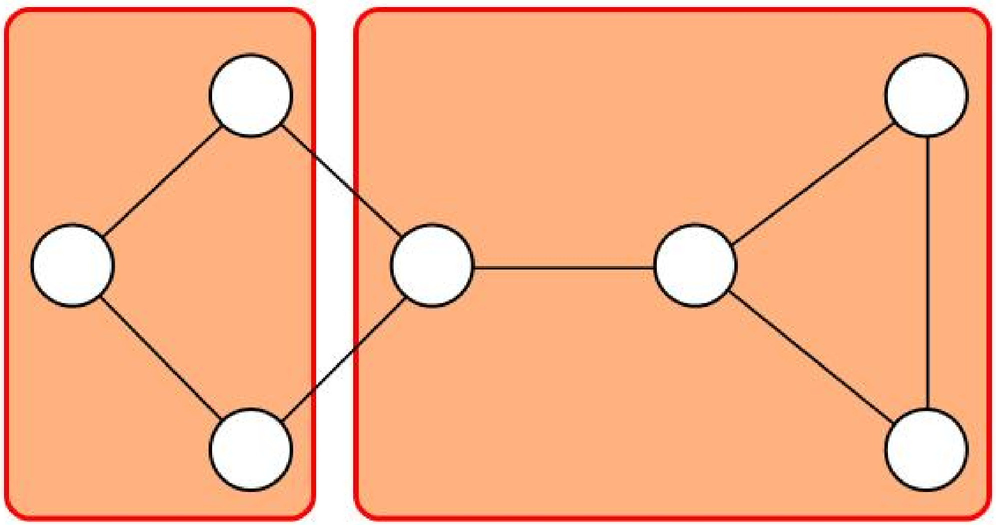


$$Q = \sum_{i=1}^{q} (e_{ii} - {a_i}^2)$$

e_{ii} = % of edges in cluster i

a_i = % of degree of nodes in cluster i

Modularity Maximization



$\frac{4}{16} - \left(\frac{6}{16}\right)^2 = -0.109$ $\qquad$ $\frac{8}{16} - \left(\frac{10}{16}\right)^2 = -0.109$

$$Q = -0.219$$

$$Q = \sum_{i=1}^{q} (e_{ii} - a_i{}^2)$$

e_{ii} = % of edges in cluster i

a_i = % of degree of nodes in cluster i

Modularity Maximization

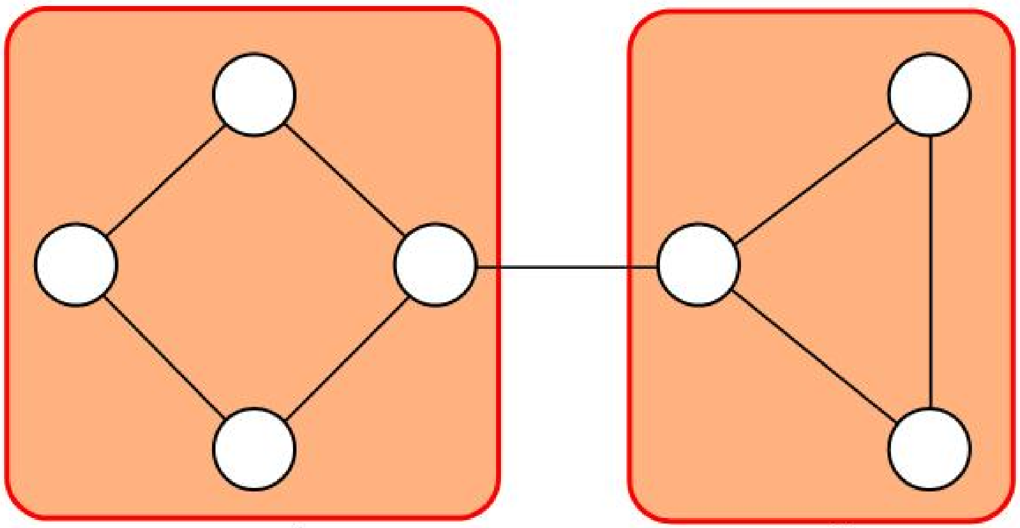
Modularity is the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random.

$$Q = \sum_{i=1}^{q} (e_{ii} - {a_i}^2)$$

High modularity

= more edges within the cluster than you expect by chance

Modularity Maximization



$$\frac{8}{16}-\left(\frac{9}{16}\right)^2=0.184$$

$$\frac{6}{16}-\left(\frac{7}{16}\right)^2=0.184$$

$$Q=0.367$$

better modularity!

$$Q=\sum_{i=1}^{q}(e_{ii}-{a_i}^2)$$

e_{ii} = % of edges in cluster i

a_i = % of degree of nodes in cluster i

Modularity Maximization

But our graph is weighted (jaccard or cosine)!

$$Q = \sum_{i=1}^{q} (e_{ii} - {a_i}^2)$$

generalize this to weighted network!

e_{ii} = % of weight of edges in cluster i
a_i = % of weight of edges of nodes in cluster i

High modularity

= more weight within the cluster than you expect by chance

Modularity Maximization

Simple Greedy Algorithm

Start:

each node is in its own cluster.

Iterate:

for each node, move it to other cluster which improve its modularity the most.

Stop when the desired total modularity is achieved or cannot be improved.

Time complexity: O(nq) per iteration - modularity gain can be computed in O(1)

In practice, it’s converge very quickly.

Modularity Maximization

Variation #2

Start:

each node is in its own cluster.

Iterate:

for each node, move it to other cluster with the highest modularity gain (could be negative).

Return the clustering with highest observed modularity.

Modularity Maximization

Variation #3 (Blondel et al.)

Start:

each node is in its own cluster.

Iterate:

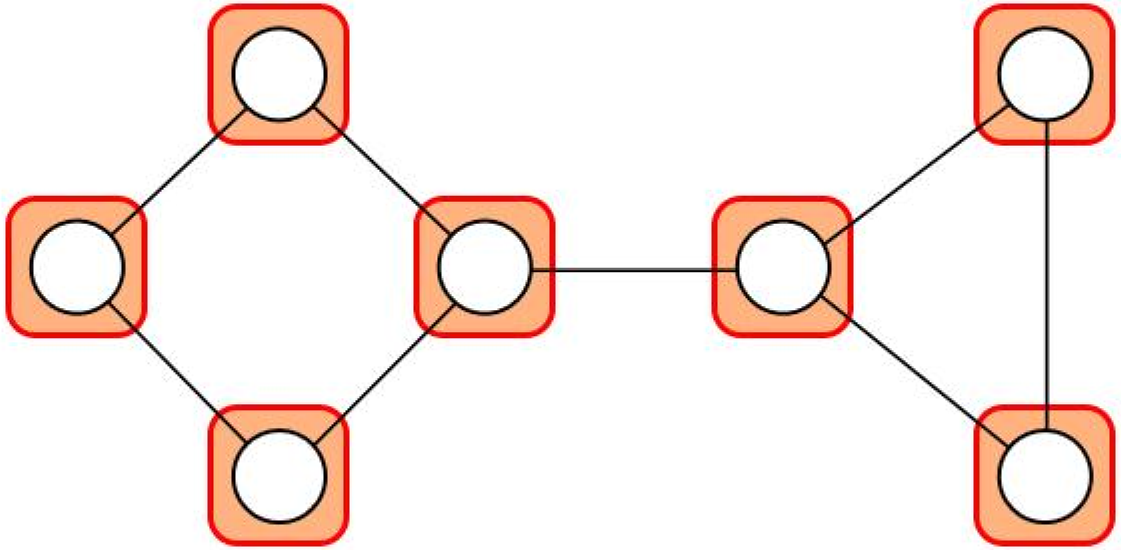
1-pass:

for each node, move it to other cluster which improve its modularity the most.

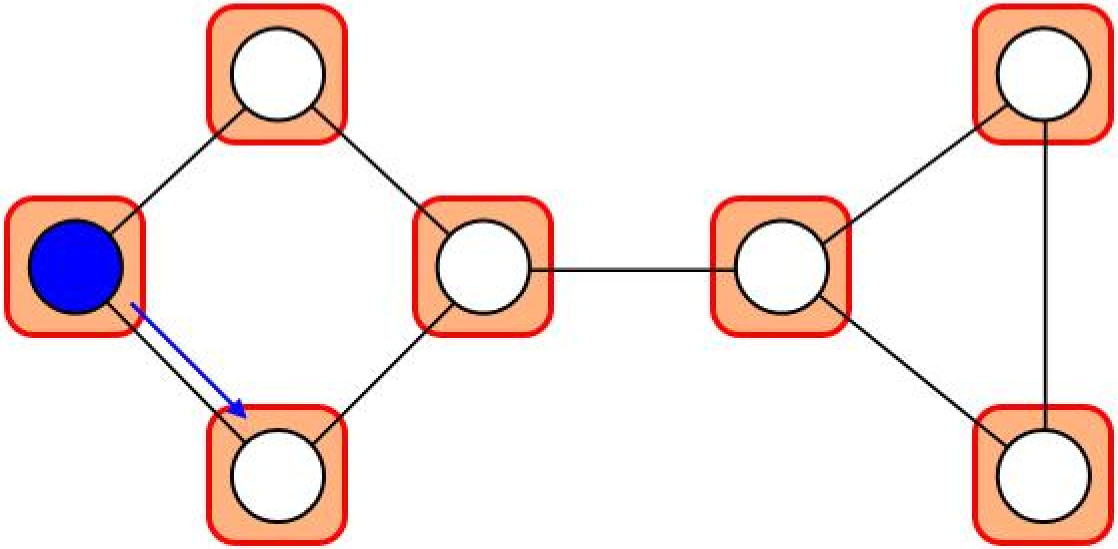
create graph G’ with each cluster (found in 1-pass) as one node, use G’ for the next iteration.

This will return a hierarchical clustering -- select the best clustering (highest modularity).

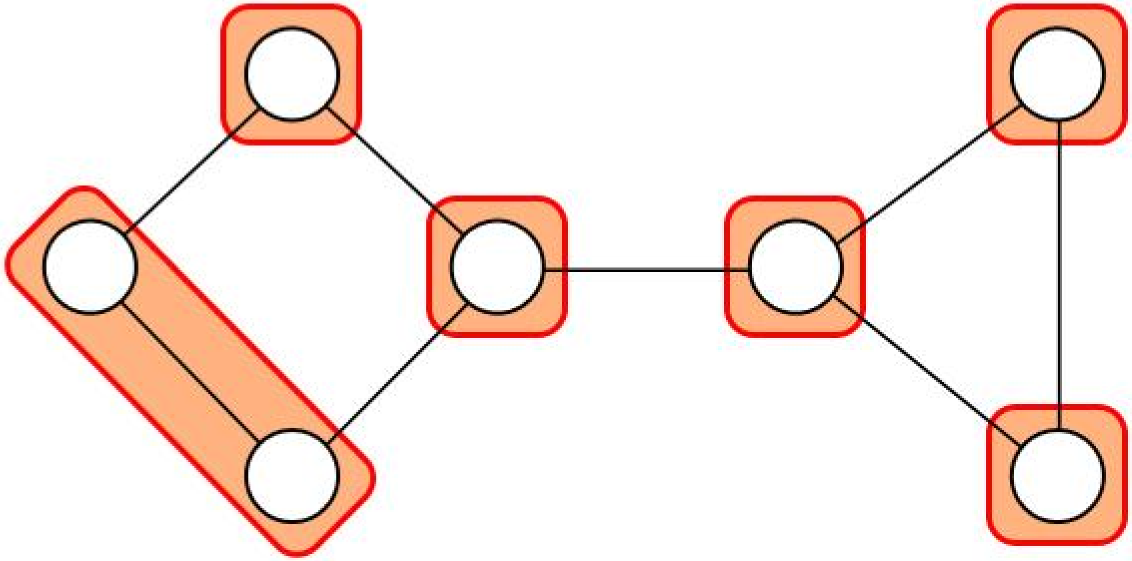
Modularity Maximization



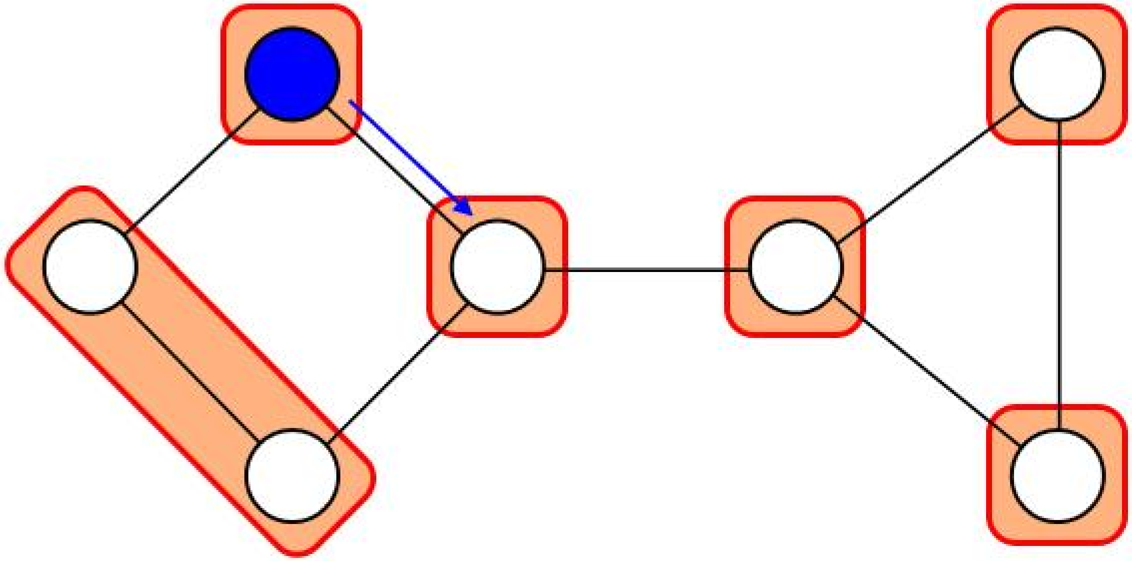
Modularity Maximization



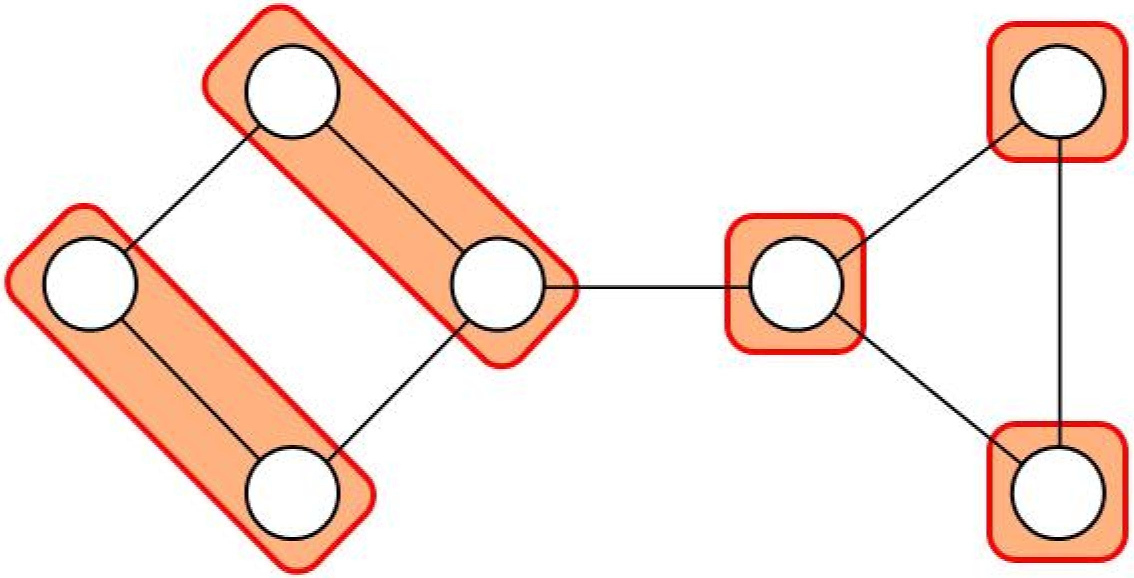
Modularity Maximization



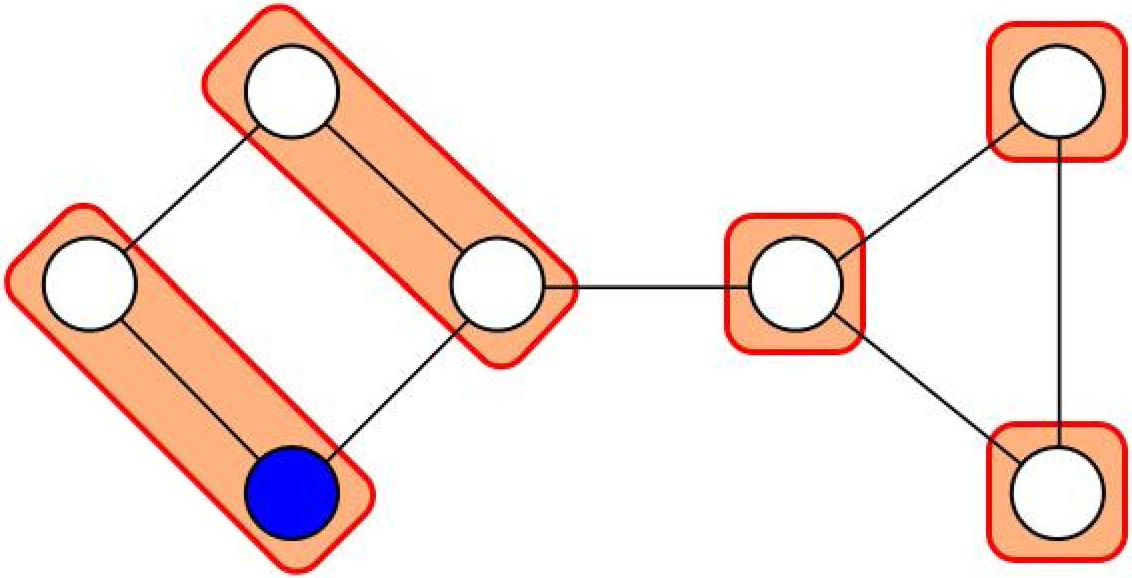
Modularity Maximization



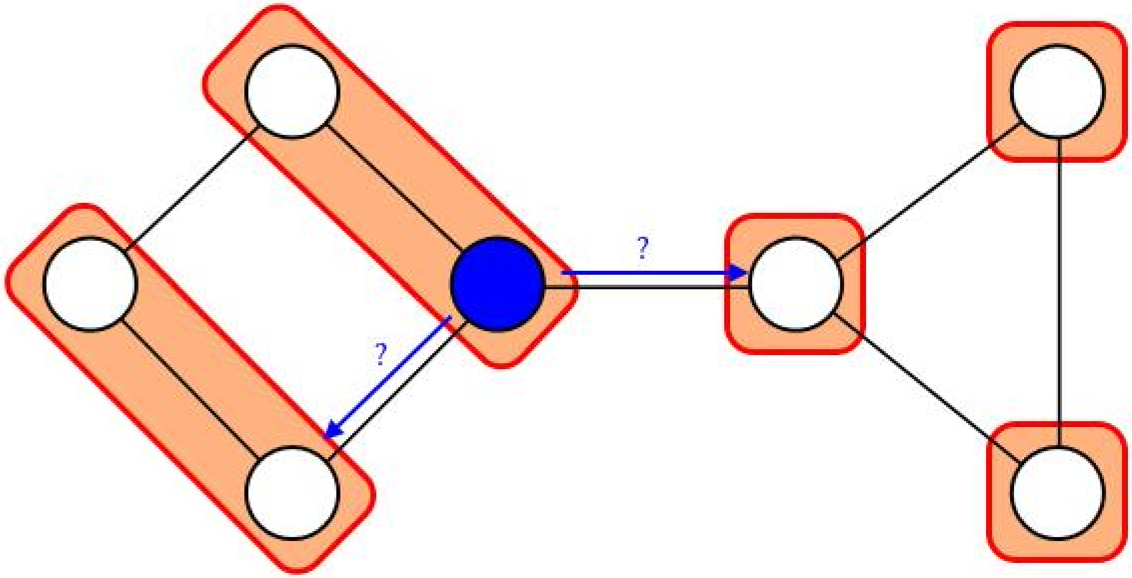
Modularity Maximization



Modularity Maximization



Modularity Maximization



… and so on

RecTree

Recursively partition the data into 2 clusters.

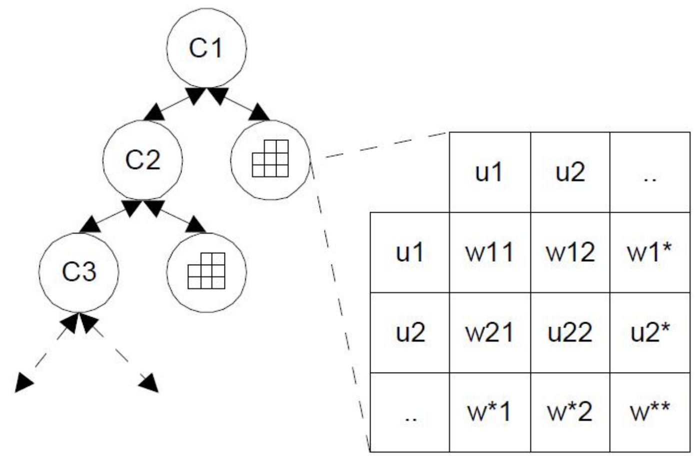
K-Means with K = 2.

Stop when:

- ➔ the partition size is small enough.
- ➔ the recursion is too deep.

O(n lg n/b), if:

- ➔ partition size = b
- ➔ recursion depth = lg n

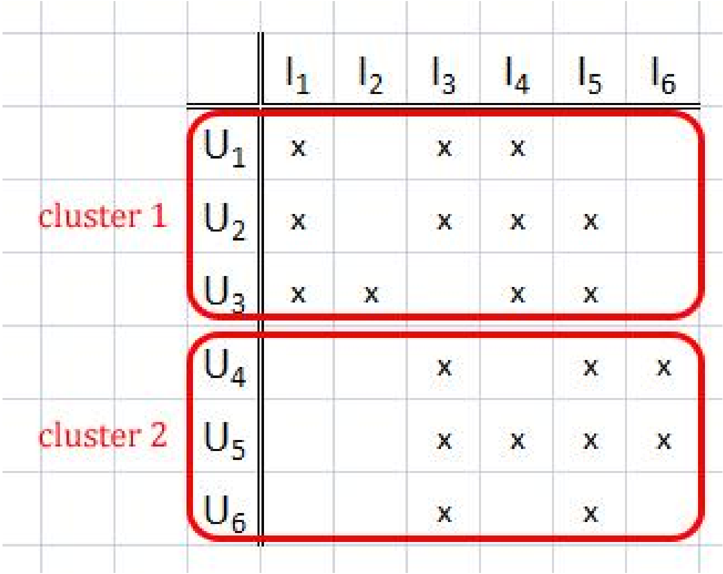


RecTree is an acronym for Recommendation Tree

Back to CF

Time complexity to build user-based clustering CF

- assume each cluster size = b
- compute similarities for one cluster = $O(b^2)$
- number of cluster = n / b
- total complexity = O(n.b)



Clustering CF

Advantage of Clustering CF

- Faster computation.
  - Small cluster size vs. entire data.

Drawback

- Researchers report that the prediction quality is lower (especially on user-based clustering CF).

Data sparsity

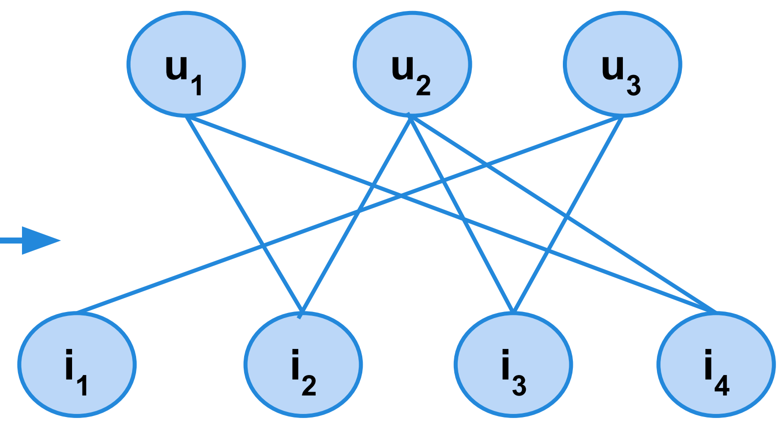
- Algorithms for sparse data
  - Graph-based method
  - Matrix factorization method
    - Also resolves:
      - First rater - new items
      - Population bias - unique taste
      - Scalability
        - Ratings can be precomputed offline
        - Parallelization is permissible
        - Rating is estimated for any unrated item in O(1) for a given user

Algorithms for sparse datasets (1)
Graph-based method (Huang et al. 2004)

- Exploit the supposed “transitivity” in user tastes
  - Example: Which item i_x could be recommended to a user u_1?

$u_1 = \{ i_2 , i_4 \}$
$u_2 = \{ i_2 , i_3 , i_4 \}$
$u_3 = \{ i_1 , i_3 \}$

	i_1	i_2	i_3	i_4
u_1	0	1	0	1
u_2	0	1	1	1
u_3	1	0	1	0



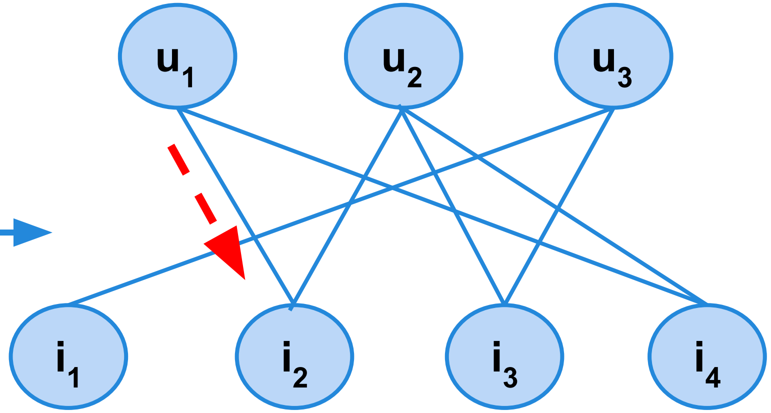
  - i_3 is recommended to u_1 because:
    - $\exists$ a three-step path between u_1 and i_3
      - $u_1 \rightarrow i_2 \rightarrow u_2 \rightarrow i_3$

Algorithms for sparse datasets (1)
Graph-based method (Huang et al. 2004)

- Exploit the supposed "transitivity" in user tastes
  - Example: Which item i_x could be recommended to a user u_1?

$u_1 = \{ i_2 , i_4 \}$
$u_2 = \{ i_2 , i_3 , i_4 \}$
$u_3 = \{ i_1 , i_3 \}$

	i_1	i_2	i_3	i_4
u_1	0	1	0	1
u_2	0	1	1	1
u_3	1	0	1	0



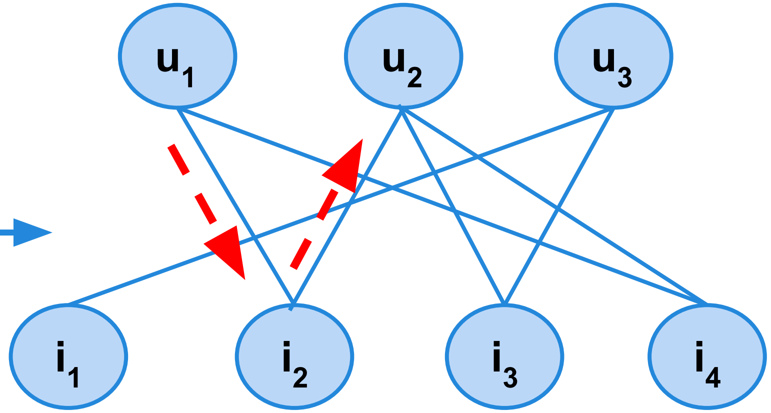
  - i_3 is recommended to u_1 because:
    - $\exists$ a three-step path between u_1 and i_3
      - u_1 -> i_2 -> u_2 -> i_3

Algorithms for sparse datasets (1)
Graph-based method (Huang et al. 2004)

- Exploit the supposed "transitivity" in user tastes
  - Example: Which item i_x could be recommended to a user u_1?

$u_1 = \{ i_2, i_4 \}$
$u_2 = \{ i_2, i_3, i_4 \}$
$u_3 = \{ i_1, i_3 \}$

	i_1	i_2	i_3	i_4
u_1	0	1	0	1
u_2	0	1	1	1
u_3	1	0	1	0



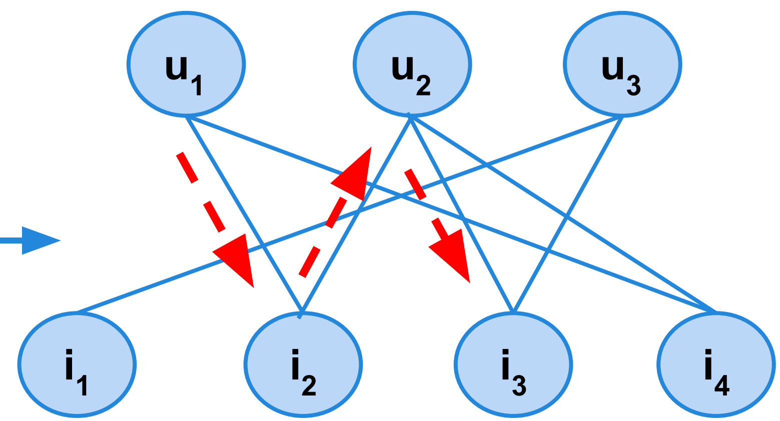
  - i_3 is recommended to u_1 because:
    - ∃ a three-step path between u_1 and i_3
      - u_1 -> i_2 -> u_2 -> i_3

Algorithms for sparse datasets (1)
Graph-based method (Huang et al. 2004)

- Exploit the supposed "transitivity" in user tastes
  - Example: Which item i_x could be recommended to a user u_1?

$u_1 = \{ i_2 , i_4 \}$
$u_2 = \{ i_2 , i_3 , i_4 \}$
$u_3 = \{ i_1 , i_3 \}$

	i_1	i_2	i_3	i_4
u_1	0	1	0	1
u_2	0	1	1	1
u_3	1	0	1	0

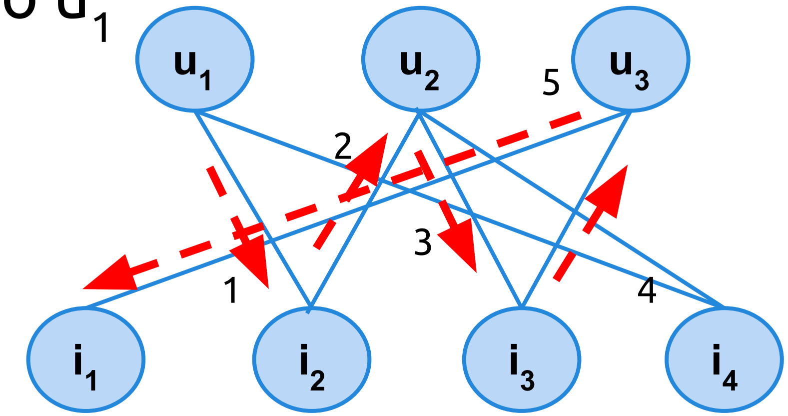


  - i_3 is recommended to u_1 because:
    - ∃ a three-step path between u_1 and i_3
      - $u_1 \rightarrow i_2 \rightarrow u_2 \rightarrow i_3$

Another 3-step path: $u_1 \rightarrow i_4 \rightarrow u_2 \rightarrow i_3$

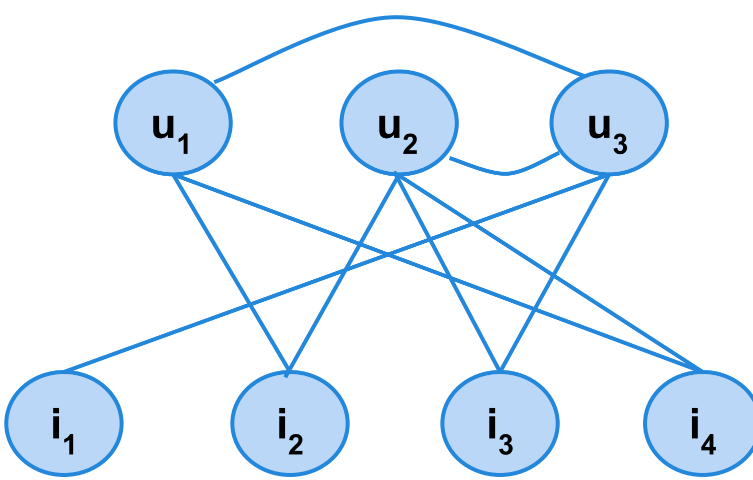
Algorithms for sparse datasets (1) Graph-based method (Huang et al. 2004)

- Consider longer paths (indirect associations) to compute recommendations in sparse matrices
  - Using path length 5, for instance
- Using path length of 3:
  - Recommend i_3 to u_1
- Using path length of 5:
  - 2 paths exist between between i_1 and u_1
  - i_1 is also recommendable to u_1



Algorithms for sparse datasets (1) Graph-based method (P. Symeonidis et al. 2011)

- Improve the relevance of recommendations
- Combining graphs
  - Unipartite graph
    - user-user
    - friendship network/ explicit social network
  - Bipartite graph
    - user-item (shown earlier)
  - Multi-modal graphs
    - friendship among users
    - user ratings on items
- Can be used by sites like Flixter
  - A community where users share film reviews and ratings



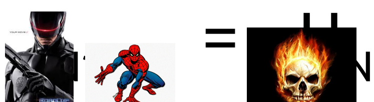
Algorithms for sparse datasets (2) Matrix Factorization - Simon Funk method

- The intuition
  - Given a list of movies that your friend have not viewed
  - How do you recommend?
    - Watch it because I watched it and liked it, **OR**
    - Match attributes (comedy, horror, ...) of movies with those attributes of other movies appreciated by friend

The Netflix 2009 $1,000,000 prize winner for the recommender's system based their solution on matrix factorization! - http://www.netflixprize.com/

Simon Funk - (Real-name: **Brandyn Webb**) independent software developer who works on Netflix prize in his spare time. He freely publishes his code...

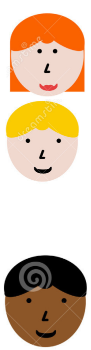
Algorithms for sparse datasets (2)
Matrix Factorization - Simon Funk method

- Factorize rating matrix
  - Define set K = {a1, a2, …, ak}, attributes of an item
    $v(i,j) \in [0,1] \ni \Sigma^{k}_{j=1} v(i, j) = 1$ for i = C, a constant
  - Recommended rating of $item_i$ for $user_j$ is:
    - $r(i,j) = U_{i\ (row)} \cdot V^{T}_{j\ (col)}$, for known r(i, j)
  - R  = $U_{N*|K|} \cdot V^{T}_{M*|K|}$

V

	a_1	a_2	…,	a_k
i_1				
i_2				
…				
i_m				

R



	i_1	i_2	…	i_m
u_1				
u_2				
…				
u_n				

=

U

	a_1	a_2	…	a_k
u_1				
u_2				
…				
u_n				

·

V^T

	i_1	i_2	…	i_m
a_1				
a_2				
…				
a_k				

Algorithms for sparse datasets (2)
Matrix Factorization - Simon Funk method

- Estimated rating of item i_{m+1} for u_x,
  - $r_{x,\, m+1} \approx U_{x\ (row)} \cdot V^T_{m+1\ (col)}$

R

	i_1	i_2	...	i_m	i_{m+1}
u_1					
u_2					
...					
u_n					

U

	a_1	a_2	...	a_k
u_1				
u_2				
...				
u_n				

V^T

	i_1	i_2	...	i_m	i_{m+1}
a_1					
a_2					
...					
a_k					

Algorithms for sparse datasets (2)
Matrix Factorization - Simon Funk method

- As such the missing ratings in R_{N*Z} can be estimated from R_{N*M} ,where M < Z:
  - $R_{N*Z} \approx U_{N*K} \cdot V^T_{Z*K}$

Analysis of matrix factorization Dimensionality reduction

- Vectors of the rating matrix, R, are of extremely high dimension
    - an item vector is an n-dimensional vector with missing user values
    - a user vector is an m-dimensional vector with missing item values
    - users and items can possibly be grouped (e.g. similar profile)
        - So can we represent users and items in smaller dimensions
        - Ideally by a constant, k
        - users and items, each represented in k dimensions

Analysis of matrix factorization Complexity

- Given a Matrix (N*M),
  - # of users = n, # of items = m
- Derive k aspect's values for m items
  - mk operations (or input: producer-defined)
- Derive k aspect's preferences for n users
  - k systems of linear eq to solve for each user
  - nCk operations, C is a constant
- Compute approximate ratings for R_{NM}
  - 2k for each rating (matrix row * col operation)
  - m * n * 2k, at most
- Time complexity O(mn)
  - Dimensionality reduction
    - Complexity reduction from $O(m^2n)$ to $O(mn)$
      - One m is "reduced" to the constant 2k :)

|items|*|users|*|Rated_items|
|items|*|aspects|*|users|*|aspects|

Google News
Collaborative Filtering in use!

- Aggregates news article from several thousand sources
- Displays them to signed-in users in a personalized way
- Collaborative filtering approach based on
  - the click history of the active user
  - the history of the larger community
- Main challenges
  - Vast amount of articles and users
  - Generate recommendation list in real-time
  - Constant stream of new items
  - Immediate reaction to the user interaction

Google News

from yr 2007 to yr 2010

- Methods
  - Two clustering techniques are used
  - Analyze history co-visits for dealing with new users
- Scalability of CF
  - Google's MapReduce technique is used for parallelization in order to make computation scalable **[Abhinandan D. et al. 2007]**
- Hybrid method
  - Combination of collaborative filtering mechanism with content-based (the next topic...)
  - Improved the quality of news recommendation and increased traffic to the site

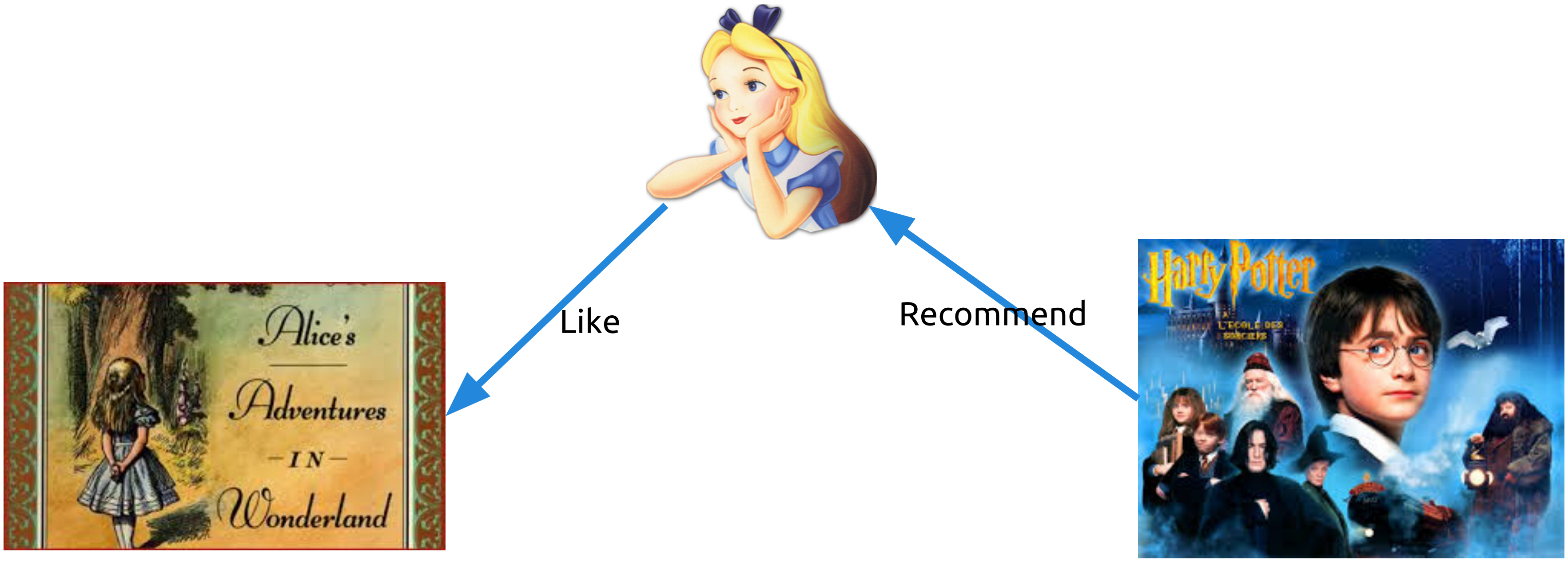
[Liu et al. 2010]

Content-Based Recommendation

Lu Bingxin & Li Jing

Why Content-based Recommendation?

- Collaborative filtering does not require any information or content about the items themselves, only using the ratings of items given by users
- It might be reasonable to exploit such information



What is Content-based Recommendation

user preferences
(such as ratings for items)

recommend items similar to what the user has liked in the past, instead of what similar users like

Content-based Recommender

relevant item(s) matching the user’s preference

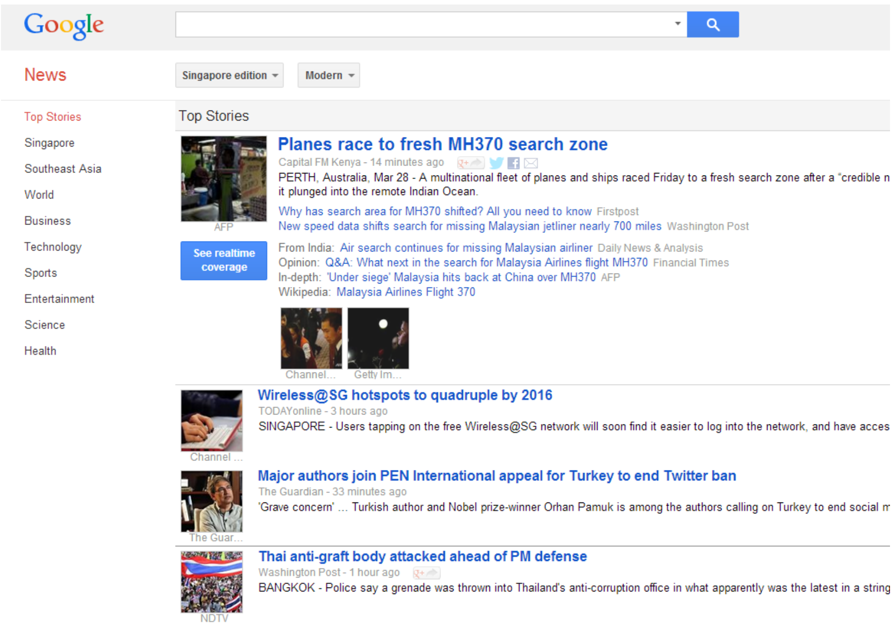
Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

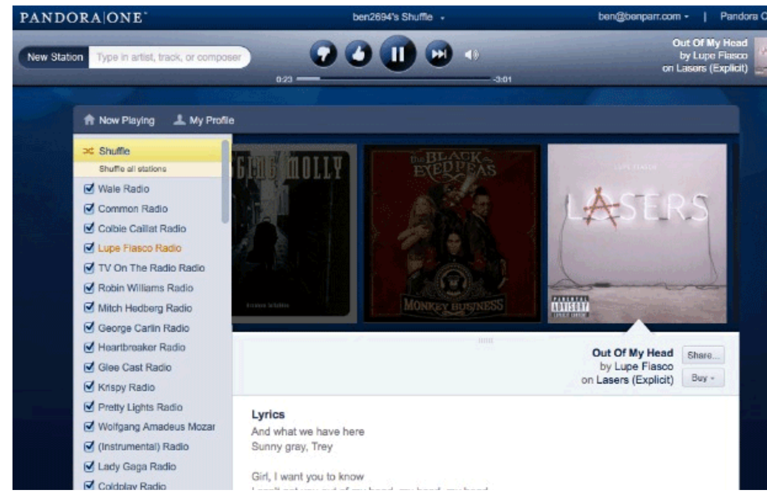
Item descriptions

not using user community information

a different form of cold-start:
require an initial description of preferences from user

Real-world example





Content-based method is often combined with collaborative filtering method, contributing to *personalize the system based on a user's interest*

Real-world example
-Panrado Radio

www.beavc.org/08presentations/pandora.ppt

Real-world example
-Panrado Radio

STEP 2: The entry is analyzed on 400 distinct musical characteristics

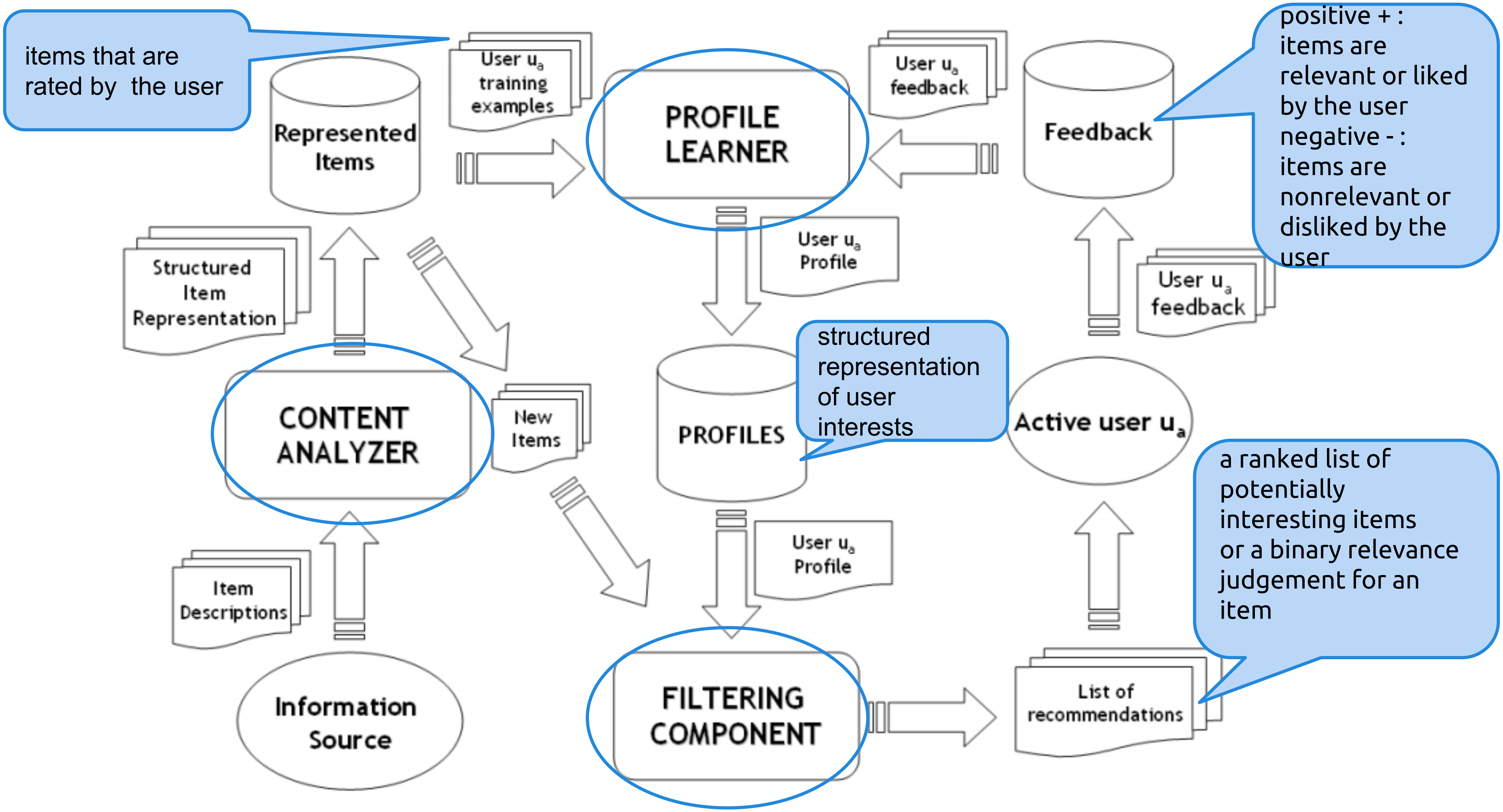
www.beavc.org/08presentations/pandora.ppt

Real-world example
-Panrado Radio

STEP 3: Similar songs are played on newly formed station

www.beavc.org/08presentations/pandora.ppt

High level architecture of a content-based recommender



http://www.ics.uci.edu/~welling/teaching/CS77Bwinter12/handbook/ContentBasedRS.pdf

Content-based recommendation as classification problem

Each item is to be classified as whether interesting to user or relevant with user preferences or not.

Two classes: positive(**+**) like/relevant ; negative(**-**) dislike/nonrelevant

items rated by the user

Training set

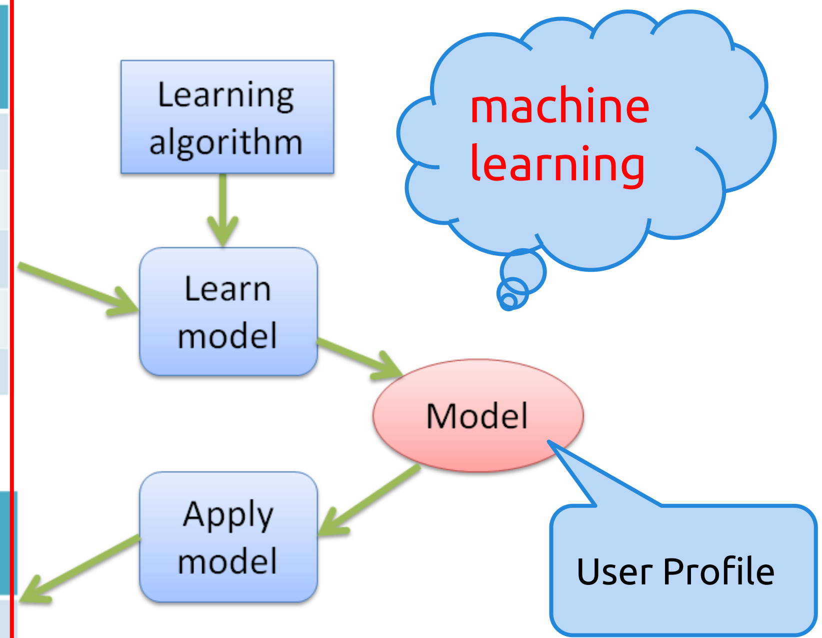
id	word1	word2	word3	...	class
doc1	0.2	0.4	3	......	+
doc2	5.3	2.5	2.7	......	+
doc3	3.2	4.6	0	......	+
[illegible]	[illegible]	2.9	3.5	......	-

item representation

not-yet-seen items

Test set

id	word1	word2	word3	...	class
doc11	0.2	0.4	3	......	?
doc12	5.3	2.5	2.7	......	?
doc13	3.2	4.6	0	......	?
doc14	1.7	2.9	3.5	......	?
......					



Item descriptions

- Some items are structured and can easily be represented by a set of attributes
  - movie
    - actor, director, genre, subject
  - book
    - title, genre, author, type, price, keyword
- Some items are unstructured text documents which have no attributes with well-defined values
  - the information source of most content-based methods
    - web pages
    - news articles
    - emails

Item Representation

- Item content
  - a set of descriptors or terms
    - typically the words that occur in a document for unstructured text

- User profile
  - often represented with the same terms as the item so that both the user profile and the items can be compared in a meaningful way

Item Representation
-for structured data

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism
...					

can maintain a list of terms (features)

Item of books

Item Representation
-for structured data

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism
...					

Item of books

same list of terms (features)

Title	Genre	Author	Type	Price	Keywords
...	Fiction, Suspense	Brunonia Barry, Ken Follett	Paperback	25.65	detective, murder, New York

Alice's User profile

Item Representation
-for structured data

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism
...					

Item of books (not yet seen by Alice)

Dice coefficient

i: a not-yet-seen item
u: user profile

$$sim(i,u) = \frac{2|keyword(i) \cap keyword(u)|}{|keyword(i)| + |keyword(u)|}$$

Title	Genre	Author	Type	Price	Keywords
...	Fiction, Suspense	Brunonia Barry, Ken Follett	Paperback	25.65	detective, murder, New York

Alice's User profile

Measure similarity between items and user profile to make recommendations

Item Representation
-for unstructured text

- A standard approach to represent unstructured document content -- Vector space model
  - selects keywords (terms) from documents
  - represent document as vector in a multi dimensional space (terms as dimensions): $d_j=\{w_{1j}, w_{2j},\ldots,w_{nj}\}$

  - *user profile* can be represented just like documents by one or more profile vectors

  - Boolean term vector
  - Weighted term vector

Item Representation
-Vector Space Model

- Boolean term vector

	team	coach	play	ball	score	game	win	lost	...
document1	1	0	1	0	0	1	0	1	
document2	0	0	0	1	1	0	1	0	
document3	1	1	1	0	0	1	0	1	
...									

- feature selection: choose only a subset of the terms in the documents

Item Representation
-Vector Space Model

- Boolean term vector

	team	**coach**	**play**	**ball**	**score**	**game**	**win**	**lost**	**...**
document1	1	0	1	0	0	1	0	1	
document2	0	0	0	1	1	0	1	0	
document3	1	1	1	0	0	1	0	1	
...									

- every word has the same relevance to a document, but it seems intuitive that
  - a word appearing more often is better suited for characterizing the document
  - a term may appear more often in longer documents

Item Representation
-Vector Space Model

- Weighted term vector
  - standard measure to weight the words: Term Frequency - Inverse Document Frequency (TF-IDF)
  - a term is assigned a weight based on
    - how often a term appears in *a particular document*
    - how frequently it occurs in *the entire document collection*

TF

Assumes that relevant terms appear more often and longer documents are not preferred to short documents

IDF

Assumes that rare terms are more relevant than frequent terms
Aims to reduce the weight of terms that appear in all documents

Weighted Term Vector
TF-IDF

- Given a term i and a document j
  - *TF(i,j)*: term frequency of keyword i in document j

the number of occurrences of keyword i in document j

$$TF(i,j) = \frac{freq(i,j)}{maxOthers(k,j)}$$

The highest number of occurrences of any other keyword k in document j

  - *IDF(i)*: inverse document frequency for keyword i

the number of all documents

$$IDF(i) = log\frac{N}{n(i)}$$

the number of documents where keyword i appears

$$TF\text{-}IDF(i,j) = TF(i,j) * IDF(i)$$

TF-IDF weight can be normalized to fall in [0,1] interval

Example TF-IDF Representation

Instead of a vector of Boolean values, the vector for each document is represented as the computed TF-IDF weights

id	men	entered	bank	charlotte	missiles	masks	aryan	guns	witnesses	reported	silver	suv	august
seg1.txt	0.239441	0	0.153457	0.195243	0	0.237029	0	0.195243	0.237029	0.140004	0.195243	0.237029	0
seg13.txt	0	0	0	0	0	0	0	0	0	0	0	0	0
seg14.txt	0	0.192197	0	0	0	0	0	0	0	0	0	0	0.172681
seg15.txt	0	0	0	0	0	0	0	0	0	0	0	0	0.149652
seg16.txt	0	0	0	0	0	0	0	0	0	0	0	0	0
seg17.txt	0	0	0	0	0	0	0	0	0	0	0	0	0
seg18.txt	0	0.158432	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0
seg19.txt	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0.155038
seg2.txt	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0
seg20.txt	0	0.234323	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0
seg21.txt	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0
seg22.txt	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0
seg23.txt	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0
seg24.txt	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0
seg25.txt	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0
seg26.txt	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0
seg27.txt	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0
seg28.txt	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	0
seg29.txt	0	0	0	0	0	0	[illegible]	0	0	0	0	0	0.142329
seg3.txt	0	0	0	0	[illegible]	0	0	0	0	0	0	0	0
seg30.txt	0.078262	0	0	0	[illegible]	0	0	0	0	0	0	0	0
seg31.txt	0	0	0.213409	0	0	0	0.194701	0	0	0	0	0	0
seg32.txt	0	0	0	0	0	0	0	0	0	0	0	0	0

The higher the value,
a term may appear more often in
a particular document or
less often in all documents,
and thus more relevant to the
topic of the document.

http://jcsites.juniata.edu/faculty/rhodes/ida/textDocViz.html

Similarity metrics based on vector space model

- Common similarity metrics to compare two vectors $d_i = (w_{1i}, w_{2i}, \ldots\ldots, w_{ki})$, $d_j(w_{1j}, w_{2j}, \ldots\ldots, w_{kj})$:

$$sim(d_i, d_j) = \frac{d_i \cdot d_j}{|d_i| \cdot |d_j|} = \frac{\sum_k w_{ki} \cdot w_{kj}}{\sqrt{\sum_k w_{ki}^2} \cdot \sqrt{\sum_k w_{kj}^2}}$$

Cosine similarity

$$sim(d_i, d_j) = 2 \frac{d_i \cdot d_j}{|d_i|^2 + |d_j|^2} = 2 \frac{\sum_k w_{ki} \cdot w_{kj}}{\sum_k w_{ki}^2 + \sum_k w_{kj}^2}$$

Dice Coefficient

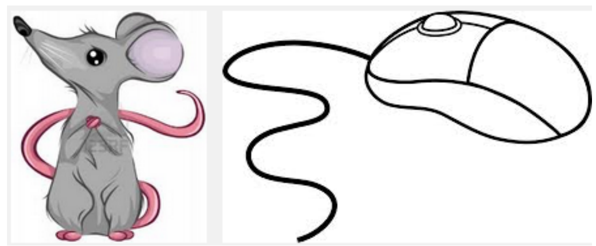
$$sim(d_i, d_j) = \frac{d_i \cdot d_j}{|d_i|^2 + |d_j|^2 - d_i \cdot d_j} = \frac{\sum_k w_{ki} \cdot w_{kj}}{\sum_k w_{ki}^2 + \sum_k w_{kj}^2 - \sum_k w_{ki} \cdot w_{kj}}$$

Jaccard coefficient

Item Representation
-More on vector space model

- Semantic meaning remains unknown
  - Polysemy

    mouse

    - The vector space model is *unable to discriminate between different meanings* of the same word
  - Synonymy

    car and vehicle

    - *No associations between different words* are made in the vector space model

Latent semantic indexing http://recommender-systems.org/latent-semantic-indexing/

Simple Method: Nearest Neighbors

- Given a set of documents D already rated by the user (like/dislike)

For each not-yet-seen item i	$m-m_1$
compute similarity between i and items in D	m_1*d
Find the N nearest neighbors of i in D	m_1
Major voting to predict ratings of i	c

- m: number of items
- m_1: number of items in D
- d: dimension of vector space
- ratings= {like, dislike}

Time complexity: O(dm^2)

In practice, most users can only rate a much small number of items compared to m, m_1 approximates to an upper bound, time complexity can approach **O(dm)**

Probabilistic Methods

Simple approach:

- 2 classes: 1/0
- simple Boolean document representation
- calculate probability that document is labeled 1/0 based on Bayes theorem

P(Label=1|X)= k*P(X|Label=1) * P(Label=1)

keywords

Doc-ID	recommender	intelligent	learning	school	Label
1	1	1	1	0	1
2	0	0	1	1	0
3	1	1	0	0	1
4	1	0	1	1	1
5	0	0	0	1	0
6	1	1	0	0	?

$$
\begin{aligned}
P(X|Label = 1) &= P(recommender = 1|Label = 1) \\
&\times P(intelligent = 1|Label = 1) \\
&\times P(learning = 0|Label = 1) \\
&\times P(school = 0|Label = 1) \\
&= {}^{3}/_{3} \times {}^{2}/_{3} \times {}^{1}/_{3} \times {}^{2}/_{3} \approx 0.149
\end{aligned}
$$

Probabilistic Methods

keywords

Doc-ID	recommender	intelligent	learning	school	Label
1	1	1	1	0	1
2	0	0	1	1	0
3	1	1	0	0	1
4	1	0	1	1	1
5	0	0	0	1	0
6	1	1	0	0	?

$$
\begin{aligned}
&P(X|Label = 1) \\
&\quad = P(recommender = 1|Label = 1) \\
&\quad \times P(intelligent \quad = 1|Label = 1) \\
&\quad \times P(learning \quad = 0|Label = 1) \\
&\quad \times P(school \quad = 0|Label = 1) \\
&\quad = {}^{3}/_{3} \times {}^{2}/_{3} \times {}^{1}/_{3} \times {}^{2}/_{3} \approx 0.149
\end{aligned}
$$

For each unlabeled item m-m_1

for each component d

compute the prior probability m_1

Overall time complexity: $O(dm^2)$

In practice: $O(dm)$

Other classification algorithms

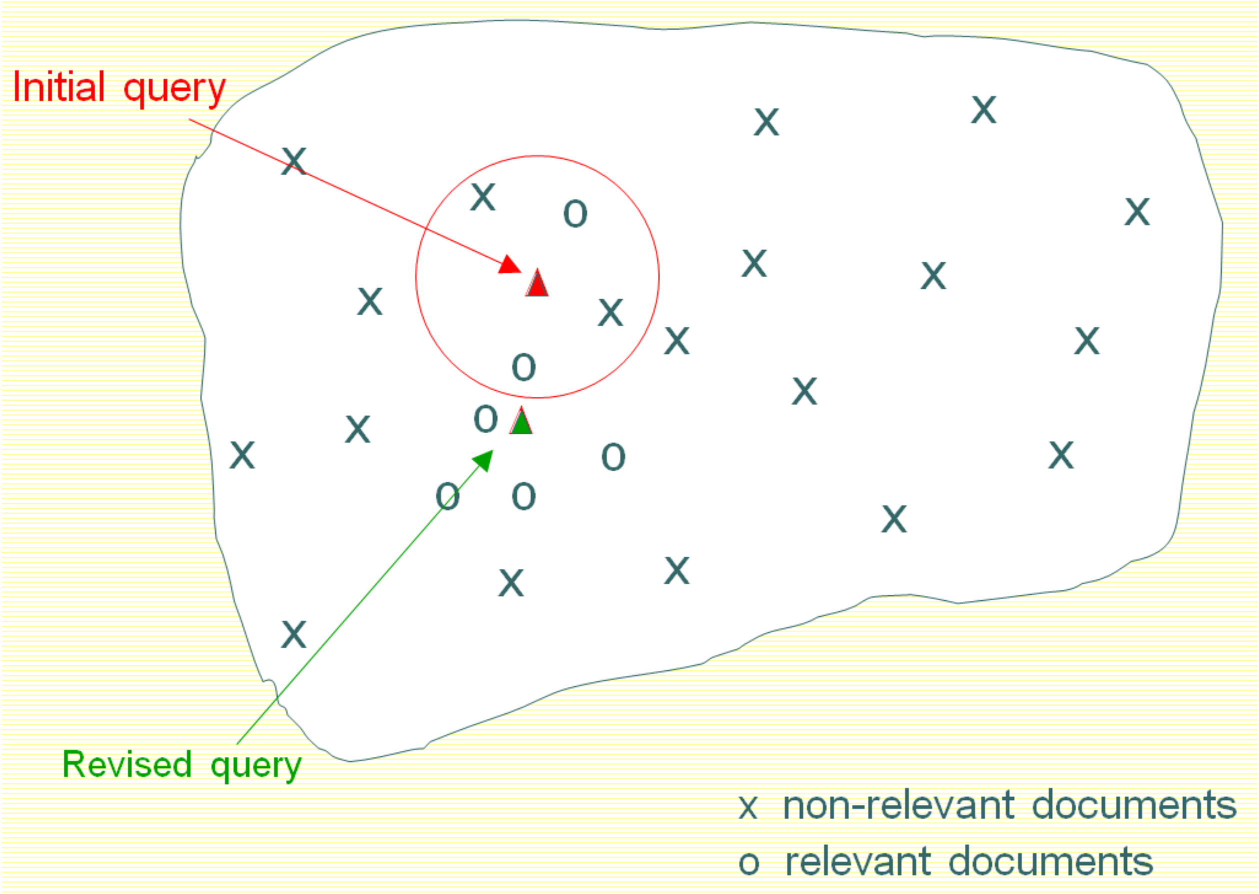
- Decision tree
- Rule induction
- Support vector machines
- Neutral network
- etc..

Relevance Feedback

- Take advantage of user relevance judgments in the retrieval process:
  - User issues a (short, simple) query and gets back an initial hit list
  - User marks hits as relevant or non-relevant
  - The system computes a better representation of the information need based on this feedback
  - Single or multiple iterations

- Idea: you may not know what you’re looking for, but you’ll know when you see it

Picture of Relevance Feedback



Rocchio Algorithm

- Query and documents are represented by TF-IDF criteria.
- Updation in practice:

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

q_m = modified query vector;
q_0 = original query vector;
α, β, γ: weights (hand-chosen or set empirically);
D_r = set of known relevant doc vectors;
D_{nr} = set of known irrelevant doc vectors

New query

Moves toward relevant documents, but away from irrelevant documents

Rocchio Algorithm: Number Example

$$\begin{aligned}\text{query vector} = \alpha \cdot \text{original query vector} \\ + \beta \cdot \text{positive feedback vector} \\ - \gamma \cdot \text{negative feedback vector}\end{aligned}$$

query	0	4	0	8	0	0
positive feedback	2	4	8	0	0	2
negative feedback	8	0	4	4	0	16

Rocchio Algorithm: Number Example

$$\text{query vector} = \alpha \cdot \text{original query vector}$$
$$+ \beta \cdot \text{positive feedback vector}$$
$$- \gamma \cdot \text{negative feedback vector}$$

query	0	4	0	8	0	0	$\alpha = 1$
positive feedback	2	4	8	0	0	2	$\beta = 0.5$
negative feedback	8	0	4	4	0	16	$\gamma = 0.25$

Typically $\beta > \gamma$, since positive feedback is more meaningful.

Rocchio Algorithm: Number Example

$$\text{query vector} = \alpha \cdot \text{original query vector} + \beta \cdot \text{positive feedback vector} - \gamma \cdot \text{negative feedback vector}$$

query	0	4	0	8	0	0	$\alpha = 1$		0	4	0	8	0	0
positive feedback	2	4	8	0	0	2	$\beta = 0.5$	+	1	2	4	0	0	1
negative feedback	8	0	4	4	0	16	$\gamma = 0.25$	-	2	0	1	1	0	4

Typically $\beta > \gamma$, since positive feedback is more meaningful.

Rocchio Algorithm: Number Example

$$\text{query vector} = \alpha \cdot \text{original query vector} + \beta \cdot \text{positive feedback vector} - \gamma \cdot \text{negative feedback vector}$$

query	0	4	0	8	0	0	$\alpha = 1$		0	4	0	8	0	0
positive feedback	2	4	8	0	0	2	$\beta = 0.5$	+	1	2	4	0	0	1
negative feedback	8	0	4	4	0	16	$\gamma = 0.25$	-	2	0	1	1	0	4
							Typically $\beta > \gamma$, since positive feedback is more meaningful.		-1	6	3	7	0	-3
new query							Negative term weights become 0.		0	6	3	7	0	0

Rocchio Algorithm

- Initial query can start with boolean vector
- Negative weights are usually ignored
- Rocchio based relevance feedback improves both recall and precision
- For reaching high recall, many iterations are needed
- Empirically determined values for the balancing weights:

  $$\alpha = 1 \quad \beta = 0.75 \quad \gamma = 0.15$$

- Positive feedback is usually more valuable than negative feedback:

  $$\beta > \gamma$$

Shortcomings of Relevance Feedback

- Relevance Feedback does not work when:
  - The users do not have sufficient initial knowledge
    - (misspelled query, ambiguous vocabulary, …)
  - There exist several prototypes of relevant documents
    - query has disjunctive answer sets (“the pop star that worked at KFC”)
    - query concerns an instance of a general concept (felines, cat)
    - documents are gathered into subsets each using a different vocabulary
- Practical problem: refining leads to longer queries that need more time to process

Relevance Feedback and the Web

Few web IR systems use relevance feedback

- hard to explain to users
- users are mainly interested in fast retrieval (i.e. no iterations)
- users usually are not interested in high recall

Nowadays: clickstream-based feedback (which links are clicked on by users)

→ implicit feedback from the writer rather than feedback from the reader

Knowledge-Based Recommendation

Suman Sourav

Why do we need knowledge based recommendation?

- Products with low number of available ratings

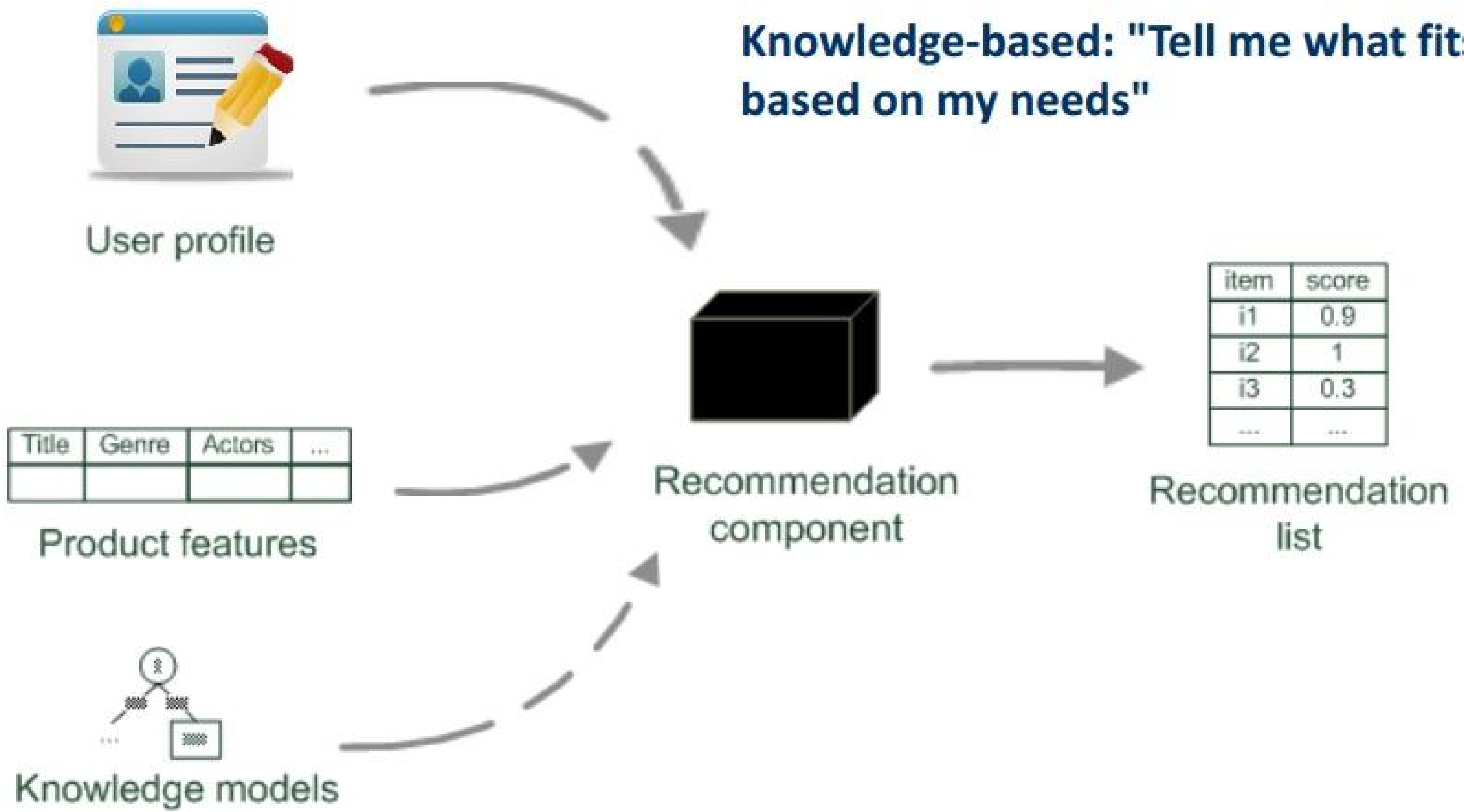
 

- Time span plays an important role
  - Five-year-old ratings for computers
  - User lifestyle or family situation changes

- Customers want to define their requirements explicitly
  - “The color of the car should be black

Knowledge based recommendation

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



ActiveBuyersGuide

digital camera product advisor

Find by: Product Use | Product Features

I need photo quality high enough for... More Info

- 5" x 7" prints (2 megapixels)
- 8" x 10" prints (4 megapixels)
- 11" x 14" prints (6 megapixels)
- No preference

My camera should fit inside a... More Info

- Shirt pocket
- Waist pack
- Backpack
- No preference

I prefer cameras that have an Epinions.com rating of

at least --select--

GET RESULTS

I want to spend... More Info

From $ up to $

I want to zoom in on subjects across a... More Info

- Small room (8 ft. away)
- Living room (15 ft. away)
- Backyard (35 ft. away)
- No preference

My preferred brands... More Info

select all that apply

- Canon
- Fujifilm
- Kodak
- Nikon
- Olympus
- Sony

more brands...

MORE GUIDANCE GET RESULTS

camcorder product advisor

Find by: Product Use | Product Features

I need a camcorder for... More Info

- Occasional & casual recordings
- Home and vacation movies
- Business productions
- No preference

I want to zoom in on subjects across a... More Info

- Playground (40 ft. away)
- Tennis court (60 ft. away)
- Park (80 ft. away)
- No preference

I prefer camcorders that have an Epinions.com rating of

at least --select--

GET RESULTS

I want to spend... More Info

From $ up to $

My camcorder should fit inside a... More Info

- Shirt pocket
- Waist pack
- Backpack
- No preference

My preferred brands... More Info

check all -- clear all

- Canon
- JVC
- Panasonic
- Samsung
- Sony

more brands...

MORE GUIDANCE GET RESULTS

mp3 player product advisor

Find by: Product Use | Product Features

My MP3 player (Digital Music Player) needs to be compatible with a... More Info

select all that apply

- Windows operating system
- Mac operating system

I want my MP3 player to hold... More Info

- A handful of songs (less than 128 MB)
- A few dozen songs (128 MB - 512 MB)
- Hundreds of songs (512 MB - 5 GB)
- Thousands of songs (5 GB or more)
- No preference

I prefer MP3 players that have an Epinions.com rating of

at least --select--

GET RESULTS

I want to spend... More Info

From $ up to $

My preferred brands... More Info

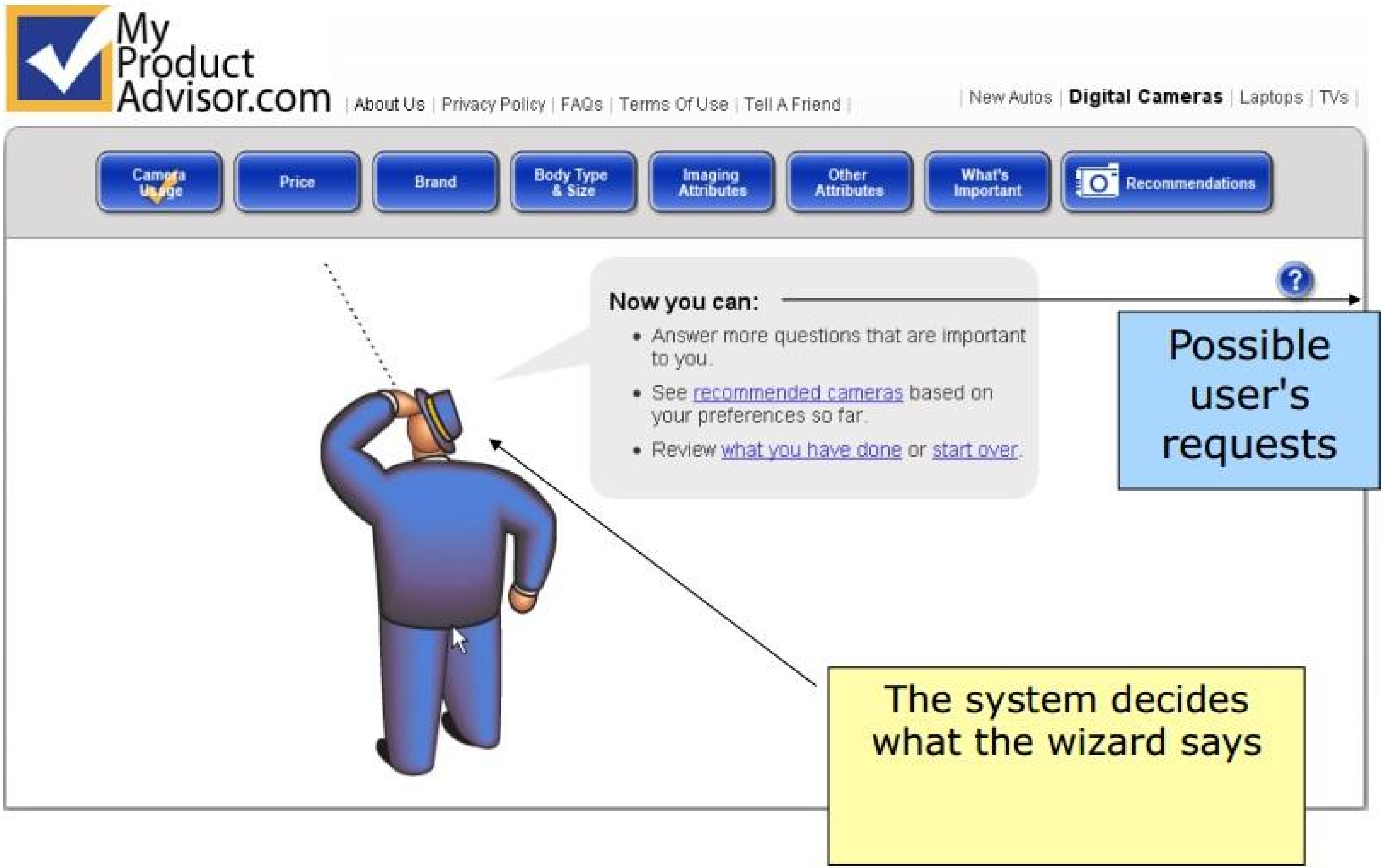
check all -- clear all

- Apple/iPod
- Creative Labs
- iRiver
- Lexar
- RCA
- Rio

more brands...

MORE GUIDANCE GET RESULTS

Wizard: My Product Advisor



Incredible India

TRAVEL

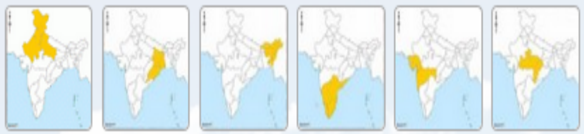
India offers a different aspect of her personality – exotic, extravagant, elegant, eclectic -- to each traveller to the country. In this section, we aim to help you choose that particular experience which will shape your vision of the country.

Rural Tourism
Mountain Trains & Luxury Trains
Eco Tourism
MICE
All Destinations
Search...

हिन्दी में देखें
Tour packages & Inquiries

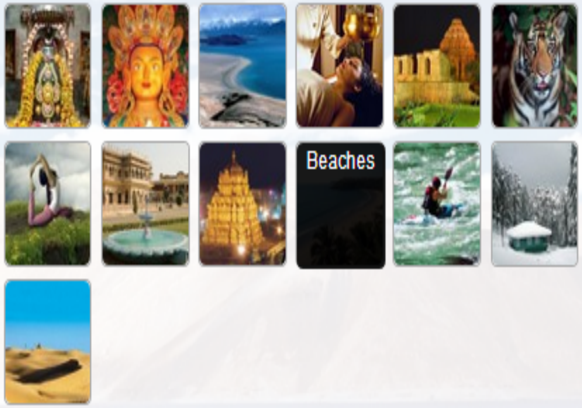
For a personal discovery of India

Choose a **REGION:**



or enter name of a **DESTINATION:**

and choose an **INTEREST:**



VACATION COACH .COM ™
TRAVEL PLANNING AS UNIQUE AS YOU

Someplace Similar.

Now pick a personality type that best describes YOU -- this will help us find similar spots based on things you like.

CULTURE CREATURE

Loves everything cultural - theater, shows, museums... local & historical culture too!

BEACH BUM

Somebody has to lay around on the beach with little umbrellas pitched in their drinks.

TRAIL TREKKER

If it's outdoors - you're there. Hiking, walking... parks, forests, mountains.

SIGHT SEEKER

Always looking for that landmark, event, or attraction.

CITY SLICKER

An urban creature who goes where the action is. Clubs, people... love the pulse of the city.

AVID ATHLETE

Always on the court or the course... always in the game... whatever game it is.

SHOPPING SHARK

Stopped looking for a cure for your shop-aholism?

WINTER WARRIOR

Will work for lift ticket. Can become quite abominable if there's no snow on the ground.

{pick one and click!}

Knowledge-based recommender systems

- **Constraint-based**
  - based on explicitly defined set of recommendation rules
  - fulfill recommendation rules

- **Case-based**
  - based on different types of similarity measures
  - retrieve items that are similar to specified requirements

- **Both approaches are similar in their conversational recommendation process**

Interacting with constraint-based recommenders

- **Conjunctive Query :**
  $\boldsymbol{\sigma_{[criteria]}(P)}$
  P: product assortment
  example: $\sigma_{[mpix \geq 10,\ price < 300]}(P) = \{p_4, p_7\}$
- The user specifies his or her initial preference
  - all at once or incrementally in a wizard-style
- The user is presented with a set of matching items
  - with explanation as to why a certain item was recommended
- The user might revise his or her requirements
  - see alternative solutions
  - narrow down the number of matching items

Constraint-based recommendation tasks

- Derive a set of recommendable items
- Find a set of user requirements such that a subset of items fulfills all constraints
  - ask user which requirements should be relaxed/modified such that some items exist that do not violate any constraint
- Find a subset of items that satisfy the maximum set of weighted constraints
- Rank items according to weights of satisfied constraints
- Provide Defaults
  - Static or Derived

Unsatisfied requirements

- **"no solution could be found"**

- **Constraint relaxation**
  - the goal is to identify relaxations to the original set of constraints
  - relax constraints of a recommendation problem until a corresponding solution has been found

- **Users could also be interested in repair proposals**
  - recommender can calculate a solution by adapting the proposed requirements

Constraint-based recommendation problem

- **Select items from this catalog that match the user's requirements**

id	price($)	mpix	opt-zoom	LCD-size	movies	sound	waterproof
P_1	148	8.0	4×	2.5	no	no	yes
P_2	182	8.0	5×	2.7	yes	yes	no
P_3	189	8.0	10×	2.5	yes	yes	no
P_4	196	10.0	12×	2.7	yes	no	yes
P_5	151	7.1	3×	3.0	yes	yes	no
P_6	199	9.0	3×	3.0	yes	yes	no
P_7	259	10.0	3×	3.0	yes	yes	no
P_8	278	9.1	10×	3.0	yes	yes	yes

- **User's requirements can, for example, be**
  - "the price should be lower than 300 $"
  - "the camera should be suited for sports photography"

Dealing with unsatisfied requirements

Suppose,

REQ = {r_1: price<=150, r_2 : opt-zoom=5×, r_3 : sound=yes, r_4 : waterproof=yes}

$$\sigma_{[price<=150,opt\text{-}zoom=5x,sound=yes,waterproof=yes]}(P) = \varnothing$$

This requirement is not satisfiable on the given set of products.

Dealing with unsatisfied requirements

Diagnosis

A minimal set of user requirements whose repair (adaptation) will allow the retrieval of a recommendation.

- $P = \{p_1, p_2,\dots,p_n\}$
- $REQ=\{r_1,r_2,\dots,r_m\}$
- $\sigma_{[REQ]}(P) = \varnothing$

We have to find $\Delta = \{d_1,d_2,\dots,d_k\}$

Such that $\sigma_{[REQ-di]}(P) \neq \varnothing \; \forall d_i \in \Delta$

Deal with unsatisfied requirements

Conflict set CS

A subset $\{r_1, r_2, \ldots, r_l\} \subseteq REQ$, such that $\sigma_{[CS]}(P) = \varnothing$.

A conflict set CS is minimal iff there does not exist aCS' with CS' $\subset$ CS.

The corresponding conflict sets are

$$CS_1 = \{r_1, r_2\}, CS_2 = \{r_2, r_4\} \text{ and } CS_3 = \{r_1, r_3\}$$

QuickXPlain

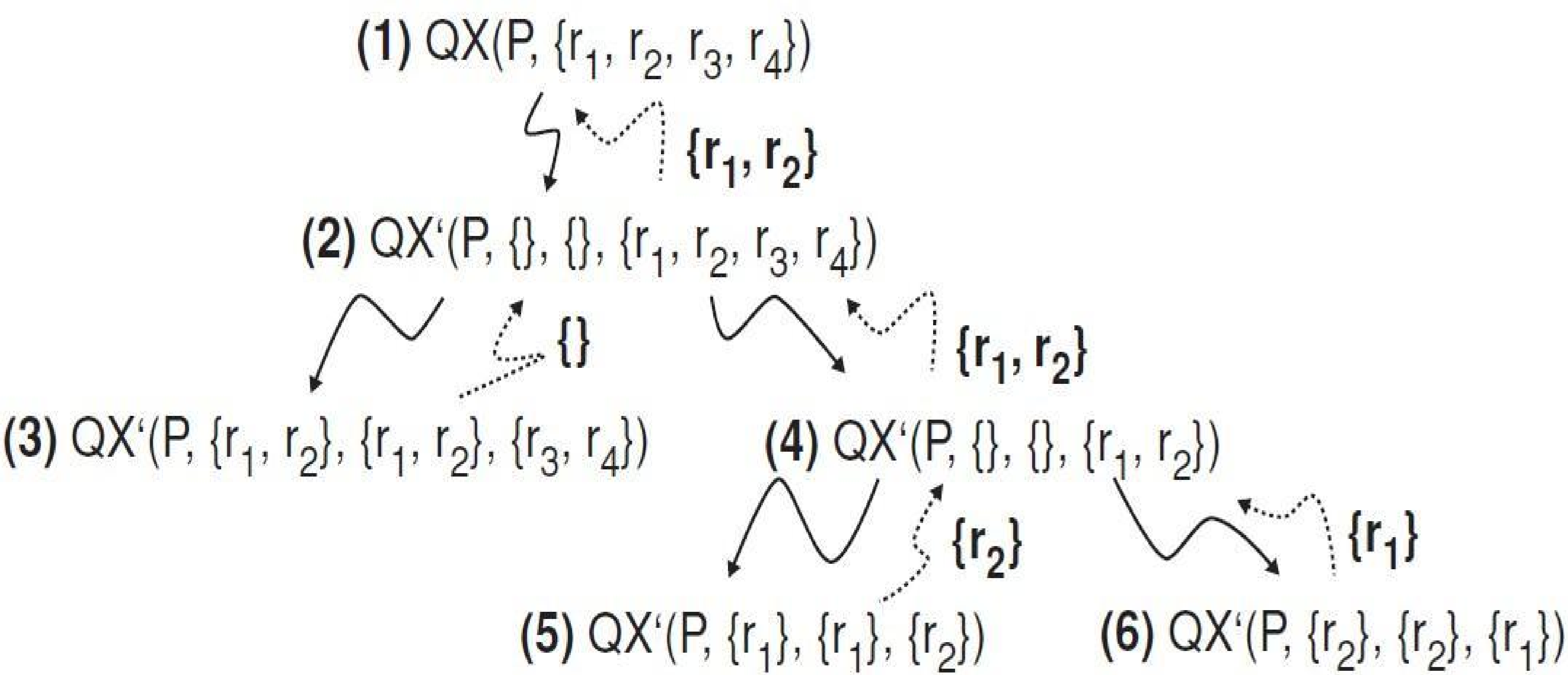
QUICKXPLAIN(P, REQ)

Input: trusted knowledge (items) P; Set of requirements REQ
Output: minimal conflict set CS
if $\sigma_{[REQ]}(P) \neq \emptyset$ or $REQ = \emptyset$ **then** return $\emptyset$
else return QX′ (P, $\emptyset$, $\emptyset$, REQ);

Function QX′(P, B, Δ, REQ)
if $\Delta \neq \emptyset$ and $\sigma_{[B]}(P) = \emptyset$ **then** return $\emptyset$;
if $REQ = \{r\}$ **then** return $\{r\}$;
let $\{r_1, \ldots, r_n\} = REQ$;
let k be $\frac{n}{2}$;
$REQ_1 \leftarrow r_1, \ldots, r_k$ and $REQ_2 \leftarrow r_{k+1}, \ldots, r_n$;
$\Delta_2 \leftarrow QX'(P, B \cup REQ_1, REQ_1, REQ_2)$;
$\Delta_1 \leftarrow QX'(P, B \cup \Delta_2, \Delta_2, REQ_1)$;
return $\Delta_1 \cup \Delta_2$;

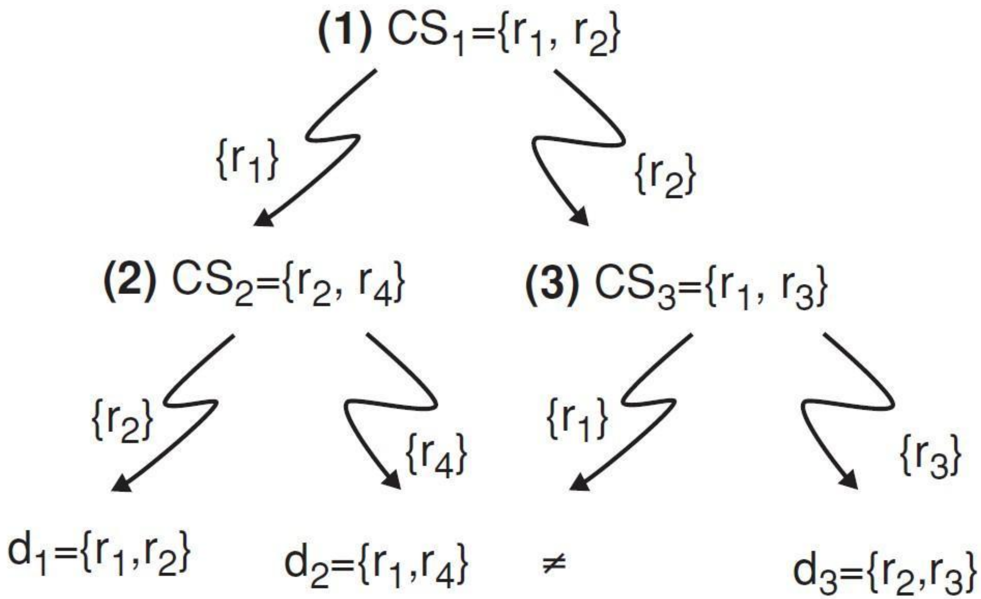
Example of QuickXPlain

- **REQ = {r1:price≤150, r2:opt-zoom=5x, r3:sound=yes, r4: waterproof=yes}**



Deal with unsatisfied requirements

- Calculate diagnoses for unsatisfied requirements



- The diagnoses derived from the conflict sets {*CS1,CS2, CS3} are {d1:{r1, r2}, d2:{r1, r4},d3:{r2, r3}}*

Repairs for unsatisfied requirements

- **Identify possible adaptations**
- **Or query the product table *P with $\pi[attributes(d)]\sigma[REQ-d](P)$***
  - $\pi[attributes(d1)]\sigma[REQ-d1](P) = \{price=278, opt\text{-}zoom=10\times\}$
  - $\pi[attributes(d2)]\sigma[REQ-d2](P) = \{price=182, waterproof=no\}$
  - $\pi[attributes(d3)]\sigma[REQ-d3](P) = \{opt\text{-}zoom=4\times, sound=no\}$

repair	price(€)	opt-zoom	sound	waterproof
Rep_1	278	10×	√	√
Rep_2	182	√	√	no
Rep_3	√	4×	no	√

Case-based Approach

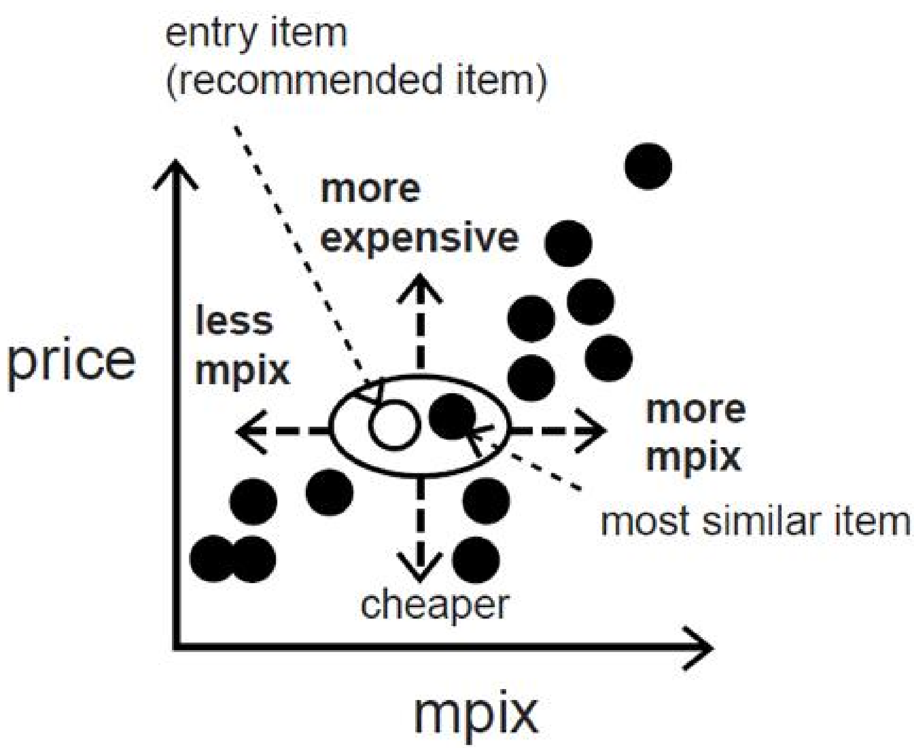
Items are retrieved based on similarity.

Critiquing

User specify their change requests that are not satisfied by the recommended item.

e.g.,

- → "lower price"
- → "more pixel"

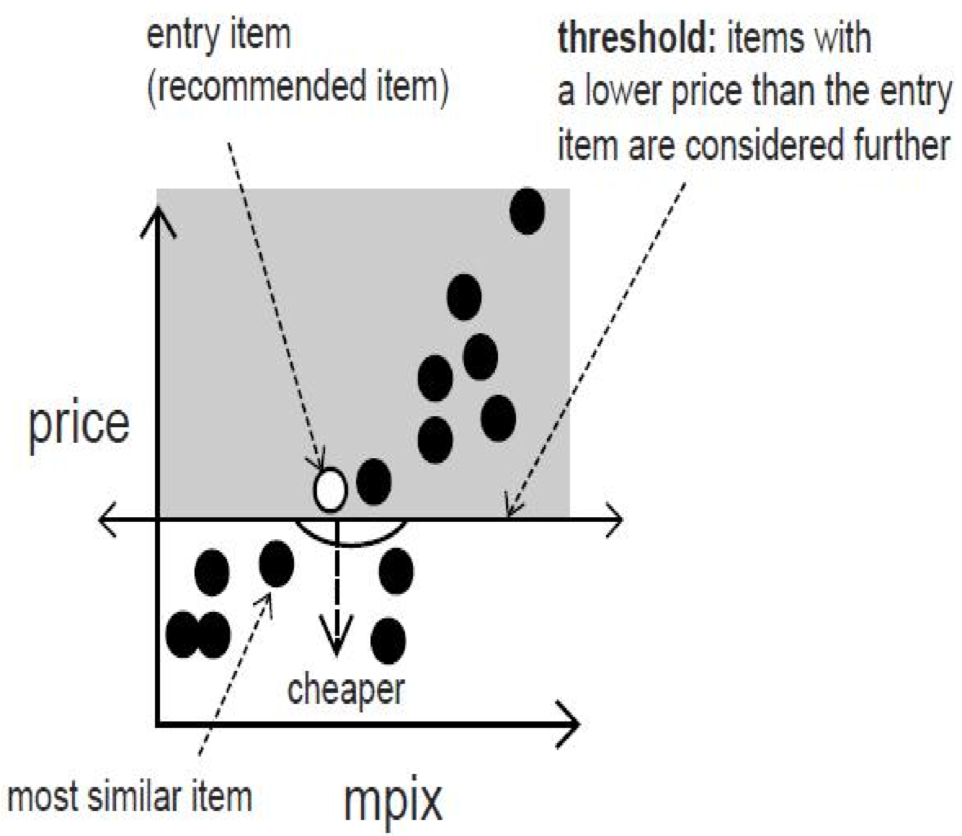


Case-based Approach

Items are retrieved based on similarity.

Critiquing
User specify their change requests that are not satisfied by the recommended item.

e.g.,
- ➔ "lower price"
- ➔ "more pixel"



Conclusion & Summary

Conclusion

- None of the models discussed are perfect or optimal.
- The choice of model normally depends the choice of the application.
- In practicality, for better performance combination of models are used rather than the pure form of any model.
- These systems are widely used in todays rapidly growing World Wide Web, and play a pivotal role in almost all major websites.

Summary

Collaborative Filtering

- “wisdom of the crowd”
- User-based or item-based CF
- Challenges in CF
  - Scalability -- clustering
  - Data Sparsity -- graph-based, matrix factorization

Content Based Recommendation

Knowledge Based Recommendation

- interactive conversational style
- based on explicit user choice only

References

- A. Das, M. Datar, A. Garg, S. Rajaram. Google news personalization: scalable online collaborative filtering. 2007
- J. Liu, P. Dolan, E. R. Pedersen. Personalized news recommendation based on click behavior. In IUI '10, 2010.
- Newman, M. E. J. Modularity and community structure in networks. In Proceedings of the National Academy of Sciences of the United States of America. 2006.
- S. H. S. Chee, J. Han, and K. Wang. RecTree: an efficient collaborative filtering method. In Proceedings of the 3rd International Conference on DataWarehousing and Knowledge Discovery, pp. 141–151, 2001.
- U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, D. Wagner. Maximizing modularity is hard. In arXiv:physics/0608255
- U. Brandes, D. Delling, M. Gaertler, R. G orke, M. Hoefer, Z. Nikoloski and D. Wagner, On Finding Graph Clusterings with Maximum Modularity. In Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'07), 2007.
- Zanker, Markus, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2011.
- http://recommender-systems.org/content-based-filtering/
- Lops P, de Gemmis M, Semeraro G. Content-based recommender systems: State of the art and trends[M]//Recommender Systems Handbook. Springer US, 2011: 73-105.
- Khan M, Nair S. Survey of Content Based Recommendation Systems in a nutshell[J]. International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE), 2014, 3(1): pp: 24-30.
- Recommender Systems : An Introduction Dietmar Jannach etal
- Robin Burke. Knowledge-based recommender systems. 2000

Backup Slides

RecTree Algorithm

```
constructRecTree( parent, data, depth )
    create a node and link it to parent
    if size( data ) ≤ maxSize OR depth ≥ maxDepth:
        computeCorrelationMatrix( data )
    else
        call K-Means( data, k = 2 )
        for each child cluster from K-Means:
            call constructRecTree( node, cData, depth + 1 )
```

Time complexity

$O(n \lg n/b)$ -- if $maxDepth = \lg n$, and $maxSize = b$.