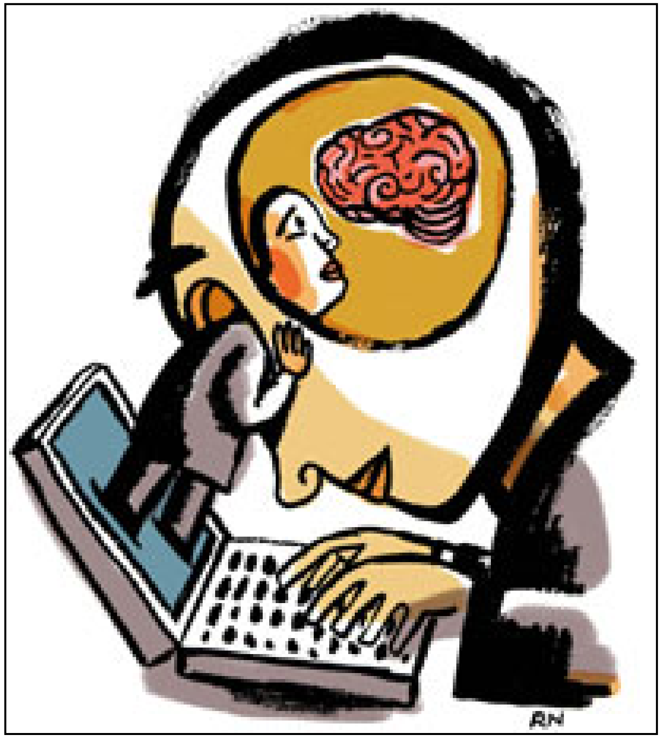
CS5344
Recommender Systems

Acknowledgement: Slides adapted from http://www.mmds.org

I Know All About You!

what you'll read next summer (Amazon, Barnes&Noble)

what movies you should watch (Reel, RatingZone, Amazon)



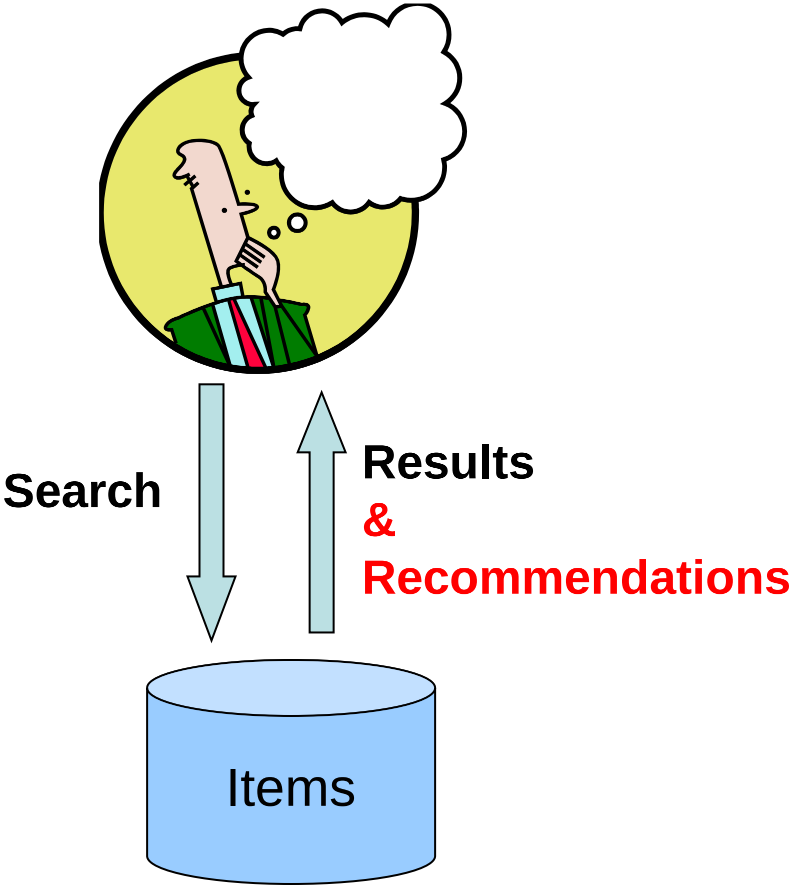
what music you should listen to (CDNow, Mubu, Gigabeat)

what websites you should visit (Alexa)

what jokes you will like (Jester)

& who you should date (Yenta)

Recommendations



Products, web sites,
blogs, news items, …

Motivation

- **Shelf space is a scarce commodity for traditional retailers**
  - e.g. Bookshelves, TV networks, movie theaters
- **Web enables near-zero-cost dissemination of information about products**
  - From scarcity to abundance
  - The Long Tail Phenomenon
- **More choice necessitates better filters**
  - Recommendation engines
  - How "**Into Thin Air**" made "**Touching the Void**" a bestseller: http://www.wired.com/wired/archive/12.10/tail.html
    - Recommenders can create demand for an obsure title

The Long Tail

- **A phenomenon whereby firms can make money by offering a near-limitless selection**
  - e.g. Netflix offers its customers a selection of over 100,000 DVD titles
  - Traditional retailers cannot offer this because of shelf space constraints

Types of Recommendations

- **Editorial**
  - List of favorites
  - Lists of “essential” items
- **Simple aggregates**
  - Top 10, Most Popular, Recent Uploads
- **Personalized**
  - Tailored to individual users
  - Amazon, Netflix

Formal Model

- *C* = set of Customers
- *S* = set of Items
- Utility Function *U*: $C \times S \rightarrow R$
  - *R* = set of ratings
  - *R* is a totally ordered set
    - e.g., **0-5** stars, real number in **[0,1]**

Customers' preferences for certain items

	Avatar	Cars	Matrix	Inside Out
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4

- Utility Matrix
  - Sparse
  - Goal of a Recommender is to predict the blanks

Key Challenges

- **Gathering “known” ratings for matrix**
  - How to collect the data in the utility matrix?
- **Extrapolating unknown ratings from known ones**
  - Mainly interested in high unknown ratings
    - Interested in what you like, and not what you do not like
- **Evaluating extrapolation methods**
  - How to measure success/performance of recommendation methods?

Gathering Ratings

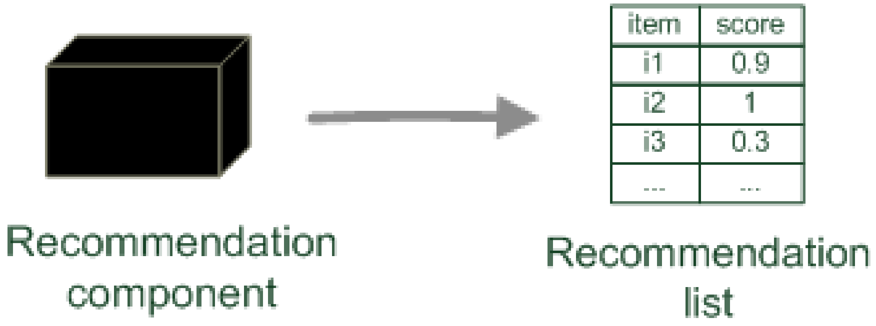
- **Explicit**
  - Ask people to rate items
  - Does not work well in practice because people cannot be bothered
- **Implicit**
  - Learn ratings from user actions
    - e.g., purchase implies high rating, clicks, time spent on some page, demo downloads, …
    - One cannot be sure whether user behavior is correctly interpreted e.g., a user may not like all books s/he has bought, the book may be a gift, …
  - What about low ratings?

Extrapolating Utilities

- **Utility matrix *U* is sparse**
  - Most people have not rated most items
  - **Cold start:**
    - New items have no ratings
    - New users have no history
- **Recommender system seen as a function**
  - Given
    - User model (e.g. ratings, preferences, demographics)
    - Items (with or without description of item characteristics)
  - Find Relevance Score
    - Used for ranking

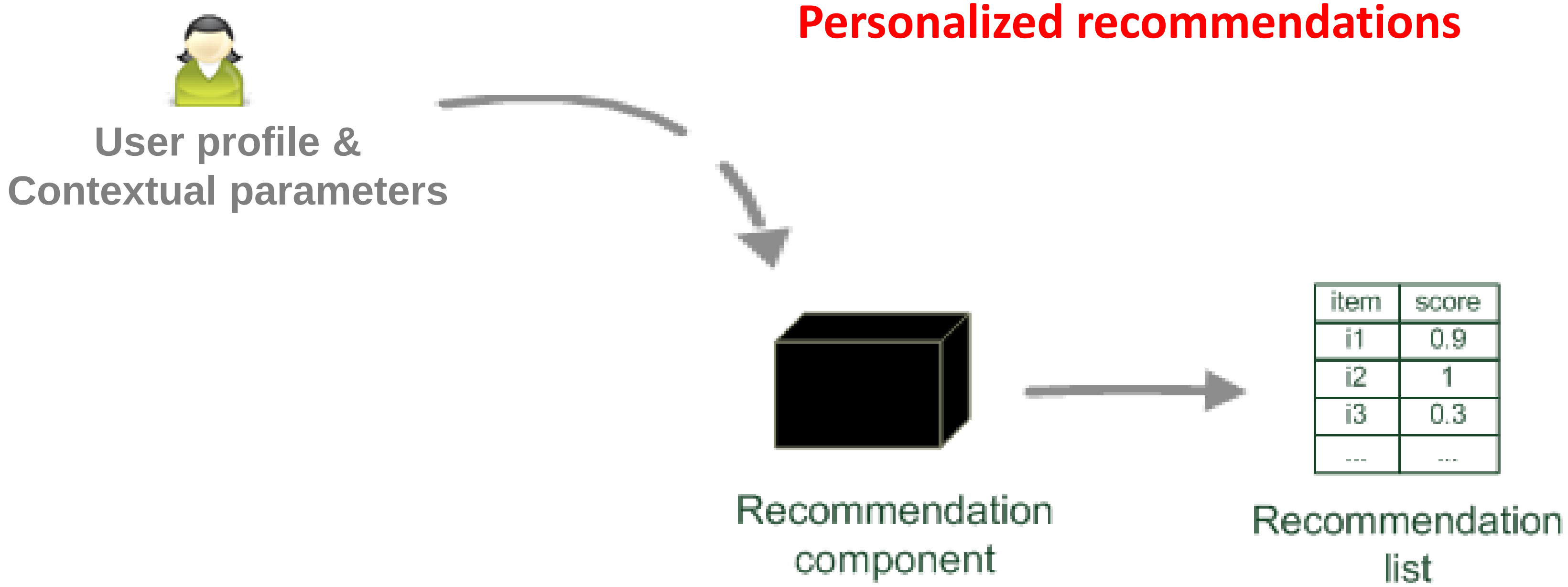
Paradigms of Recommenders

Recommender systems reduce information overload by estimating relevance



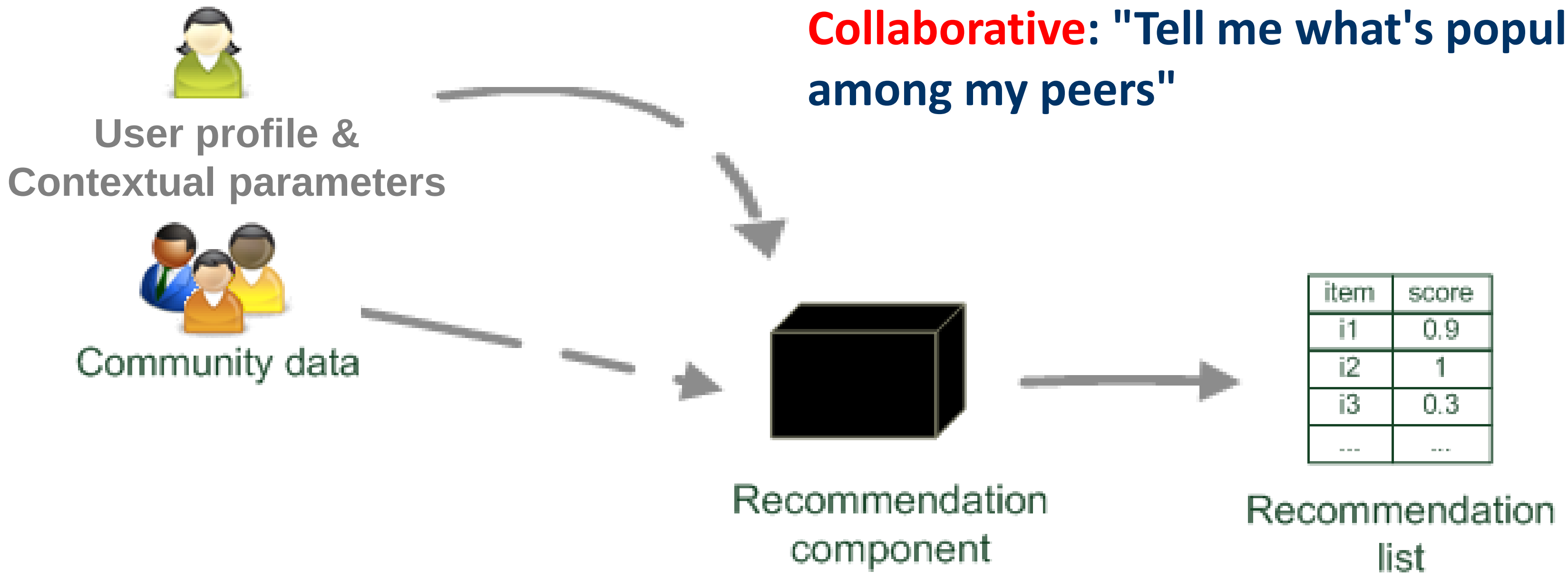
Paradigms of Recommenders

Personalized recommendations



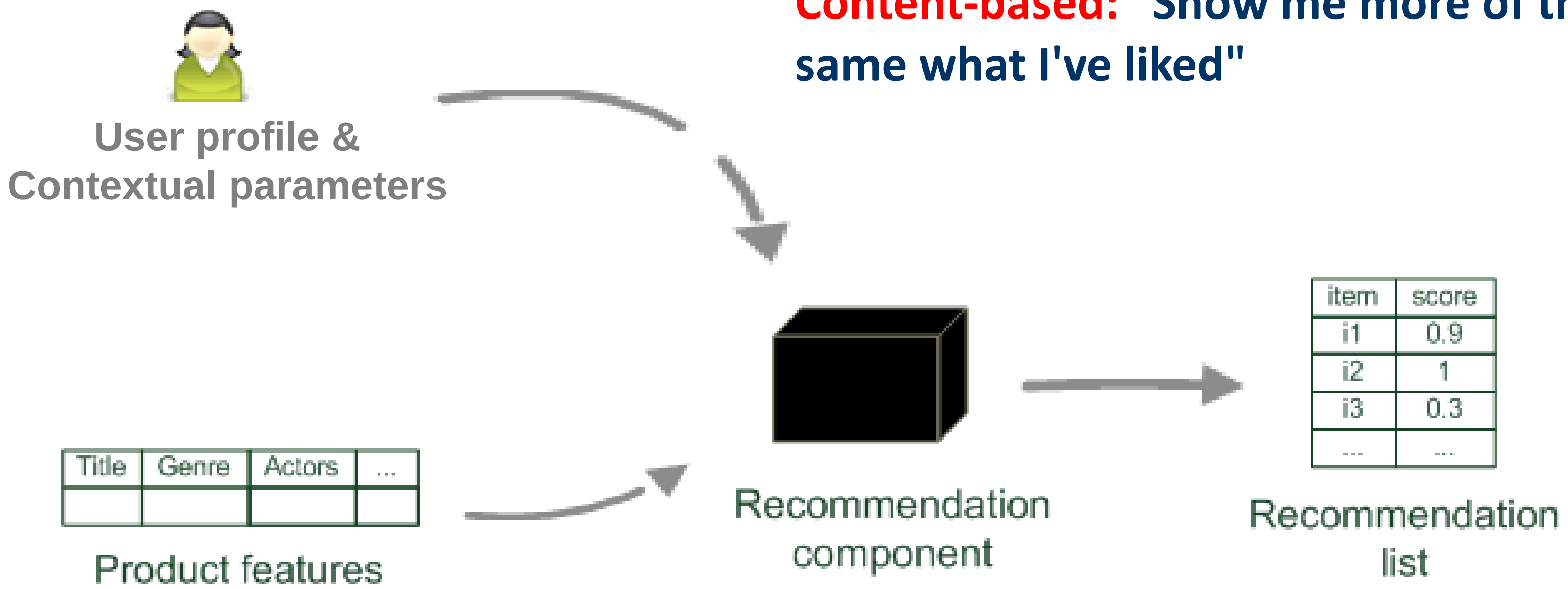
Paradigms of Recommenders

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



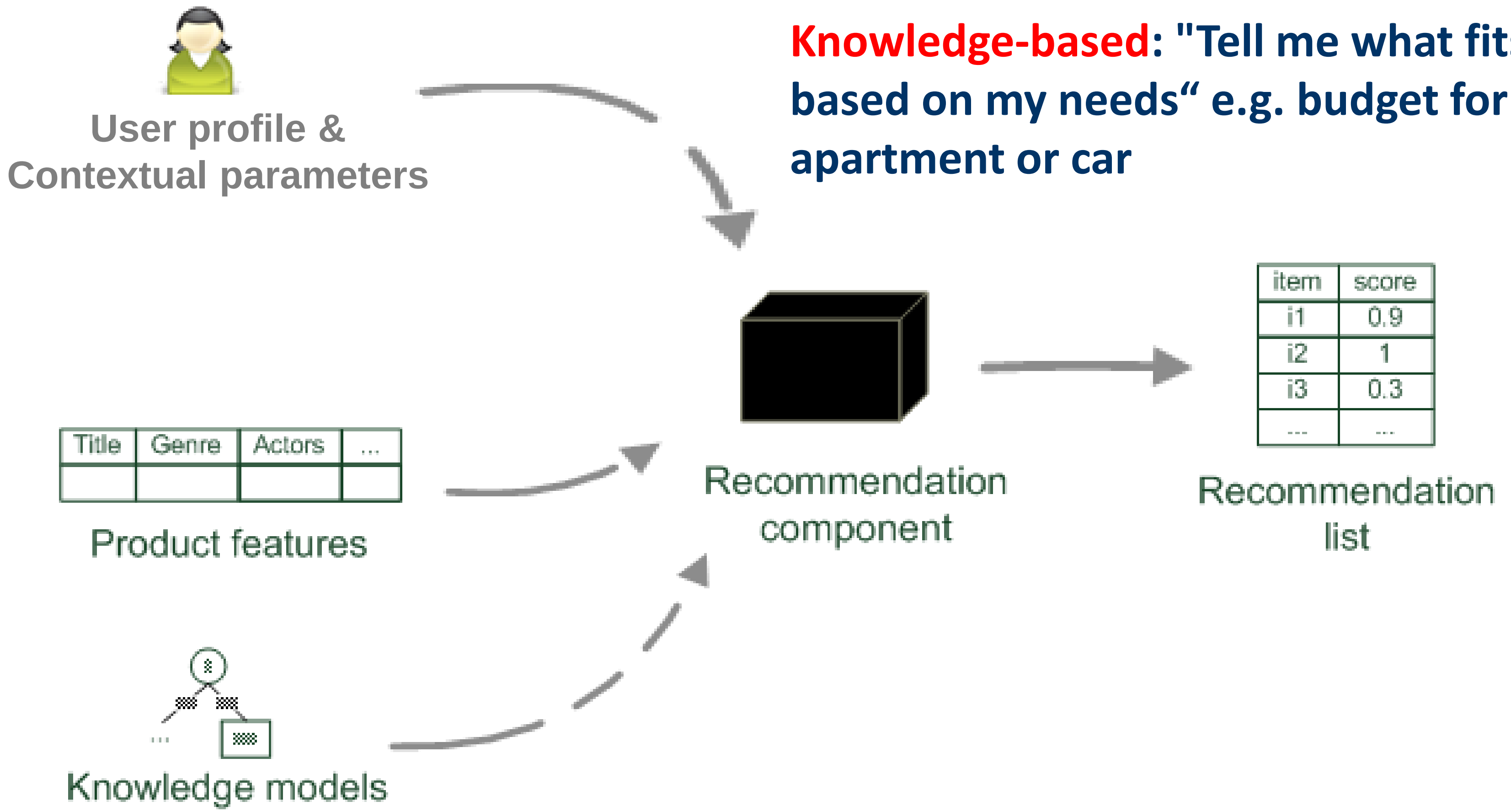
Paradigms of Recommenders

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



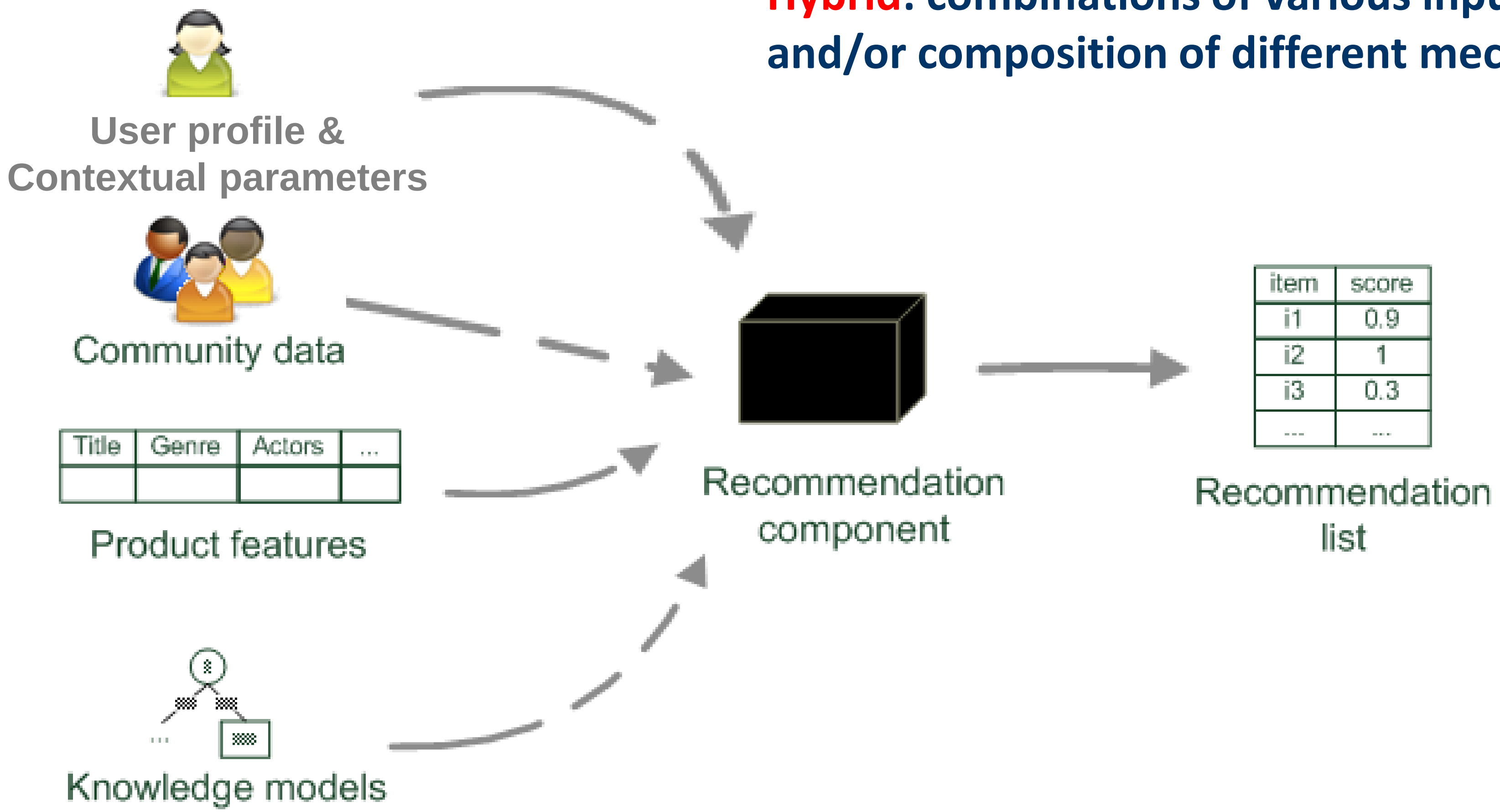
Paradigms of Recommenders

Knowledge-based: "Tell me what fits based on my needs“ e.g. budget for apartment or car



Paradigms of Recommenders

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



Content-Based Recommender Systems

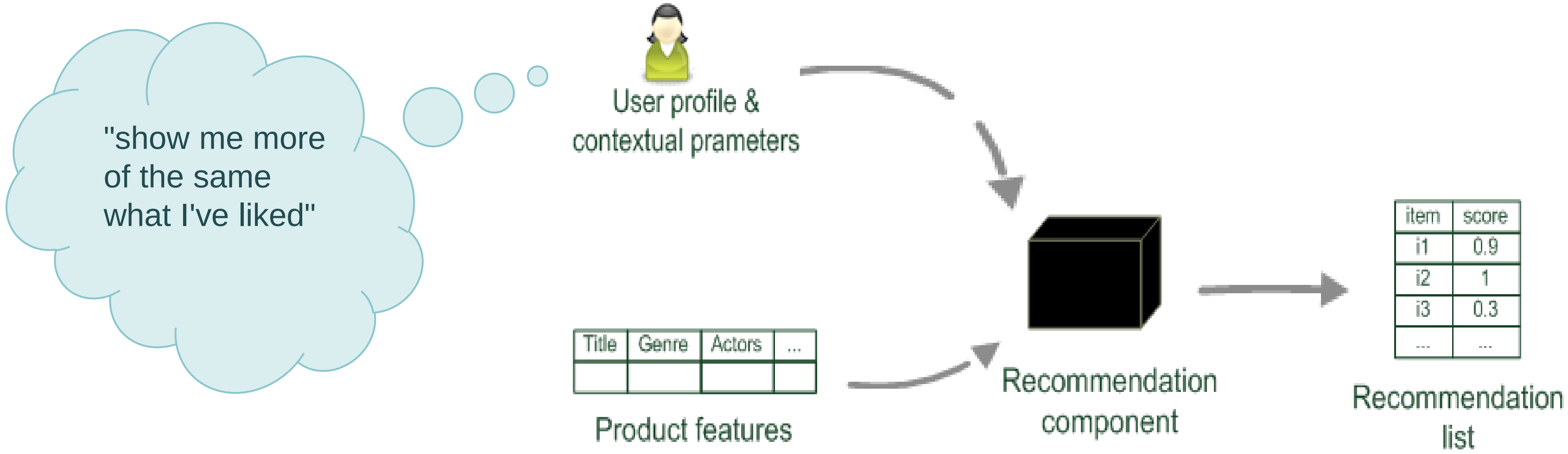
Content-Based Recommendation

Focus on Properties of Items

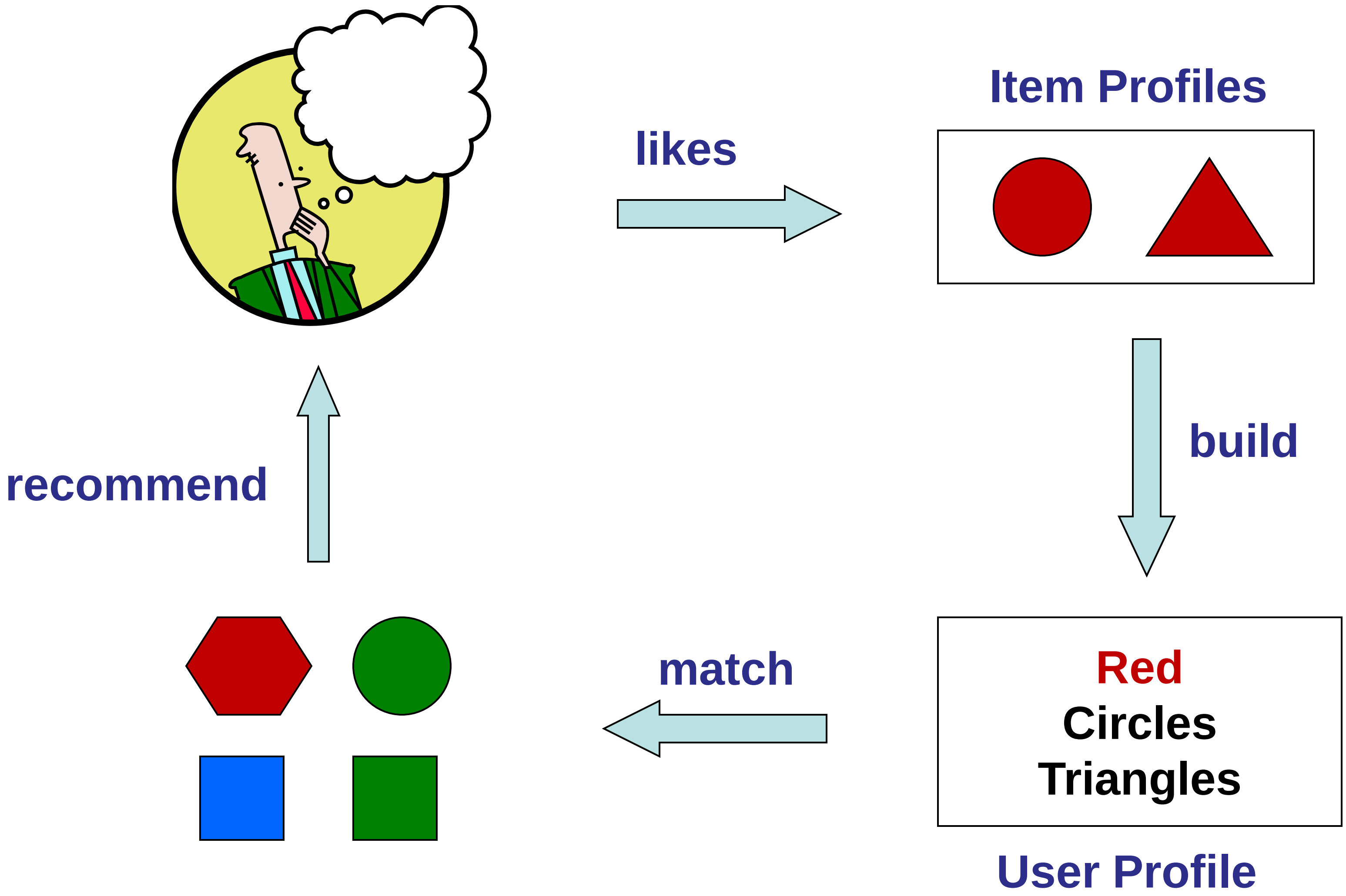
- **Main idea: recommend to customer (C) items that are similar to previous items rated highly by C**
- **Examples**
  - Movie recommendations
    - Recommend movies with same actor(s), director, genre…
  - Websites, blogs, news
    - Recommend other sites with “similar” content

Content-Based Recommendation

- **Inputs**
  - Information about available items e.g. genre ("content")
  - User profile
- **Task**
  - Learn user preferences
  - Locate/recommend items "similar" to user preferences



Plan of Action



Item Profiles

	a_1	a_2	a_3	…	a_k
i_1	1	0	1	0	0
i_2	1	0	1	0	1
i_3	1	1	0	1	1

- **For each item, create an item profile**
- **Profile is a set (vector) of features**
  - Movies: producer, title, actor(s), director,…
  - Text: set of "important" words in document
  - Vector might be boolean or real-valued
    - e.g. Items are movies, Features are actors
    - Item profile is a boolean vector where a "1" is set for the component corresponding to the actor in the movie
- **How to pick important features?**
  - Usual heuristic is **TF-IDF**
    - Term: Feature
    - Document: Item

User Profiles

	a_1	a_2	a_3	…	a_k
i_1	1	0	1	0	0
i_2	1	0	1	0	1
i_3	1	1	0	1	1
i_4	1	1	0	1	1
i_5	1	1	0	0	1

- **Vector to describe user preferences**
  - E.g. Average of rated item profiles
  - Suppose user x rated items i_2, i_3 and i_4, then we have

x	1	0.6	0.3	0.6	1

- **Predict preference of *x* for items he has not rated**
  - Given user profile ***x*** and item profile ***i***, estimate

$$utility(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$$

  - E.g. $\cos(x, i_1) = 0.55$; $\cos(x, i_5) = 0.9$

    → recommend i_5 to x

Pros: Content-based Approach

- **No need for data on other users**
  - No cold start or sparsity problems
- **Able to recommend to users with unique tastes**
- **Able to recommend new & unpopular items**
- **Able to provide explanations**
  - Can explain recommended items by listing the content-features that caused an item to be recommended

Cons: Content-based Approach

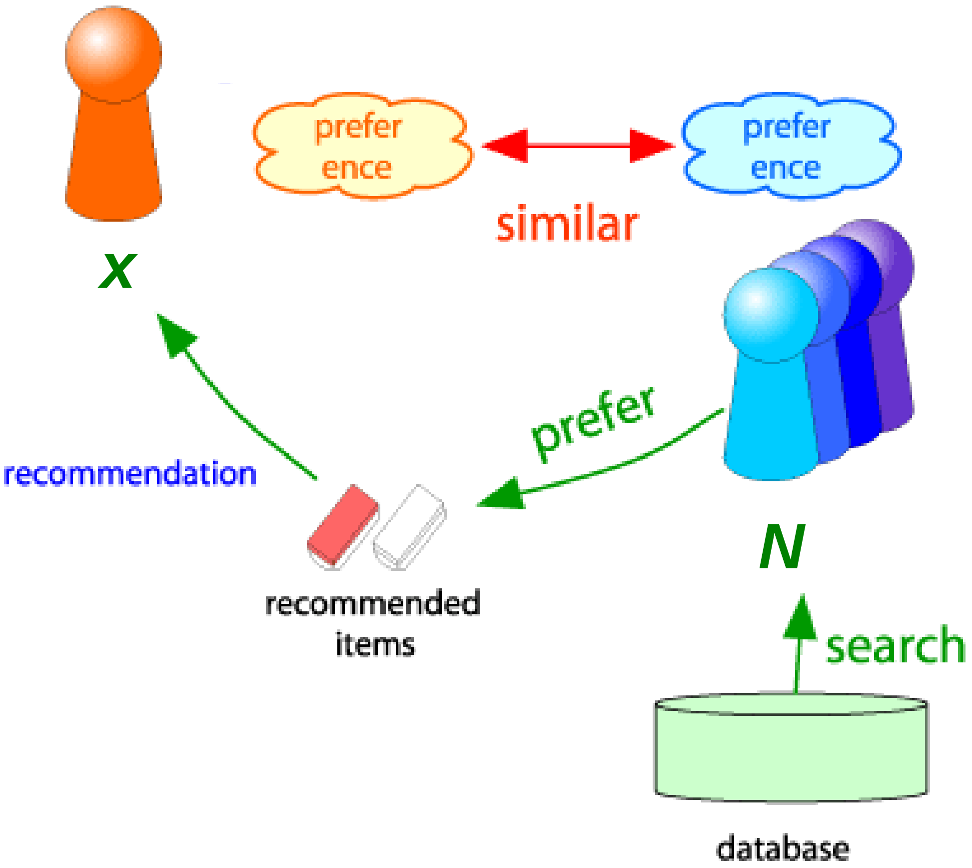
- **Finding the appropriate features is hard**
  - e.g., images, movies, music
- **Cold start problem for new users**
  - How to build a user profile?
- **Overspecialization**
  - Never recommend items outside user’s content profile
  - People might have multiple interests
- **Unable to exploit quality judgments of other users**

Collaborative Filtering

Harnessing Quality Judgments of Other Users

Collaborative Filtering

- **Consider user *x***
- **Find set *N* of other users whose ratings are "similar" to *x*'s ratings**
- **Estimate *x*'s ratings based on ratings of users in *N***



Finding "Similar" Users

r_x = [*, _, _, *, ***]
r_y = [*, _, **, **, _]

- **Let r_x be the vector of user x's ratings**
- **Jaccard similarity measure**
  - Problem: Ignores the value of the rating
- **Cosine similarity measure**
  - sim($\boldsymbol{x}$, $\boldsymbol{y}$) = cos($\boldsymbol{r_x}$, $\boldsymbol{r_y}$) = $\frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$
  - **Problem:** Treats missing ratings as "negative"
- **Pearson correlation coefficient**
  - S_{xy} = items rated by both users $\boldsymbol{x}$ and $\boldsymbol{y}$

$$sim(x, y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_x} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_y} (r_{ys} - \overline{r_y})^2}}$$

r_x, r_y as sets:
r_x = {1, 4, 5}
r_y = {1, 3, 4}

r_x, r_y as points:
r_x = {1, 0, 0, 1, 3}
r_y = {1, 0, 2, 2, 0}

*$\overline{r_x}$, $\overline{r_y}$ … avg. rating of **x**, **y***

Similarity Metric

item / user	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- **Intuitively, we want sim(A,B) > sim(A,C)**
- **Jaccard similarity:** sim(A,B) =1/5 < sim(A,C) = 2/4
  - **Ignores the value of the ratings**
- **Cosine similarity:** sim(A, B) = 0.380 > sim(A,C) = 0.322
  - **Treat missing ratings as 0, consider as "negative"/dislike**

Similarity Metric

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- **Solution: Centered Cosine similarity (Pearson Correlation)**

$$\boldsymbol{sim(x,y)} = \frac{\sum_{s\in S_{xy}}(r_{xs}-\overline{r_x})(r_{ys}-\overline{r_y})}{\sqrt{\sum_{s\in S_x}(r_{xs}-\overline{r_x})^2}\sqrt{\sum_{s\in S_y}(r_{ys}-\overline{r_y})^2}}$$

  - Normalize rating by subtracting row mean (average rating of user)
  - Turn low ratings into negative numbers and high ratings into positive numbers
  - sim(A,B) = 0.092 > sim(A,C) = -0.559

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	2/3			5/3	-7/3		
B	1/3	1/3	-2/3				
C				-5/3	1/3	4/3	
D		0					0

= 2 – (2+4+5)/3

Rating Predictions

- **From similarity metric to recommendations**
- **Predict rating of user *x* for item *i* (not yet seen by *x*)**
  - Let $\boldsymbol{r_x}$ be the vector of user ***x***'s ratings
  - Let ***N*** be the set of ***k*** users most similar to ***x*** **and** have rated item ***i***
    - Use average of their ratings $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
    - Or some weighted measures $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$

$$s_{xy} = sim(x, y)$$

User-Based Nearest-Neighbor CF

- **Ratings of user Alice, and some other users**
- **Determine whether Alice will like or dislike *Item5*, which Alice has not yet rated or seen**

	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 = 0.79

sim = 0.42

sim = 0

sim = -0.69

Compute PCC similarity of Alice and users who have rated Item 5

Let |N| = 2. Then N = {User1, User2}

Estimate rating of Alice for Item 5

$$r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi} = 4$$

Item-Based Collaborative Filtering

- **So far: User-based collaborative filtering**
- **Another view: Item-based**
  - For item ***i***, find other similar items (that has been rated by user ***x***)
  - Estimate rating for item ***i*** based on ratings (of the user) for similar items
  - Can use same similarity metrics and prediction functions as in user-based CF

$$\boxed{r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}}$$

s_{ij}… similarity of items ***i*** and ***j***
r_{xj}…rating of user ***x*** on item ***j***
N(i;x)… set of items similar to ***i*** rated by ***x***

Item-Based CF

- **Predict Alice's rating for Item5**
  - 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

Item-Based CF

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

unknown rating

rating between 1 to 5

Item-Based CF

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Estimate rating of movie **1** by user **5**

Item-Based CF

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	
1	1		3		?	5			5		4		
2			5	4			4			2	1	3	
3	2	4		1	2		3		4	3	5		sim = 0.41
4		2	4		5			4			2		sim = -0.10
5			4	3	4	2					2	5	sim = -0.31
6	1		3		3			2			4		sim = 0.59

- Use PCC to identify movies similar to movie 1 (rated by user 5)
- Let |N| = 2. Then N = {3, 6}
- Predict by taking weighted average:

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

$r_{1,5} = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6$

Collaborative Filtering

- **Pros: Works for any kind of item**
  - No feature selection needed
- **Cons:**
  - **Cold Start**
    - Need enough users in the system to find a match
  - **Sparsity**
    - Hard to find users that have rated the same items
  - **First rater**
    - Cannot recommend an unrated item
  - **Popularity bias**
    - Tend to recommend popular items

Hybrid Recommenders

- **All base techniques have their shortcomings**
  - e.g. cold start
- **Integrate different recommenders and combine predictions**
  - e.g. add content-based methods to CF
    - Item profiles for new item problem
    - Demographics to deal with new user problem

Evaluating Predictions

movies

users

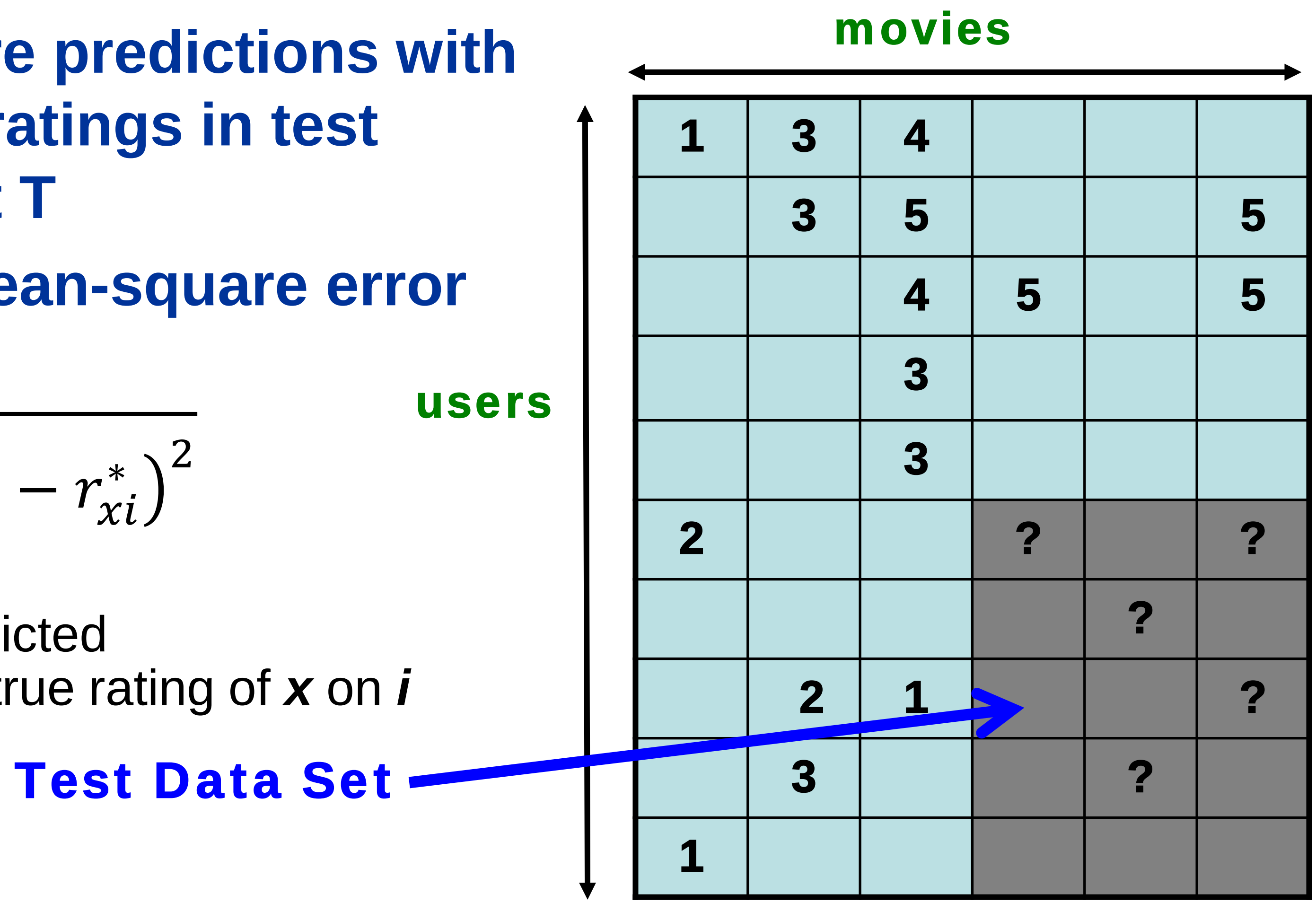
1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

Evaluating Predictions

- **Compare predictions with known ratings in test data set T**
- **Root-mean-square error (RMSE)**

$$\sqrt{\sum_{xi}\left(r_{xi} - r_{xi}^{*}\right)^{2}}$$

$\boldsymbol{r_{xi}}$ is predicted
$\boldsymbol{r_{xi}^{*}}$ is the true rating of ***x*** on ***i***



Complexity

- **Expensive step is finding *k* most similar customers**
  - **O(|C|) where C is the set of customers**
- **Too expensive to do at runtime**
  - Need to pre-compute
  - Naïve pre-computation takes time **O(k ·|C|)**
- **Ways to reduce computation**
  - Near-neighbor search in high dimensions (**LSH**)
  - Clustering
  - Dimensionality reduction

Tip: Add Data

- **Leverage all the data**
  - Don’t try to reduce data size in an effort to make fancy algorithms work
  - Simple methods on large data do best
- **Add more data**
  - e.g., add IMDB data on genres
- **More data beats better algorithms**

http://anand.typepad.com/datawocky/2008/03/more-data-usual.html

Summary

- **Recommender systems have important applications, many of which have been successful in practice**
- **Content-based recommender systems**
  - Focused on items
- **Collaborative Filtering**
  - Focused on people
  - User-based
  - Item-based
- **Hybrid methods that combine both content-based systems and collaborative filtering**