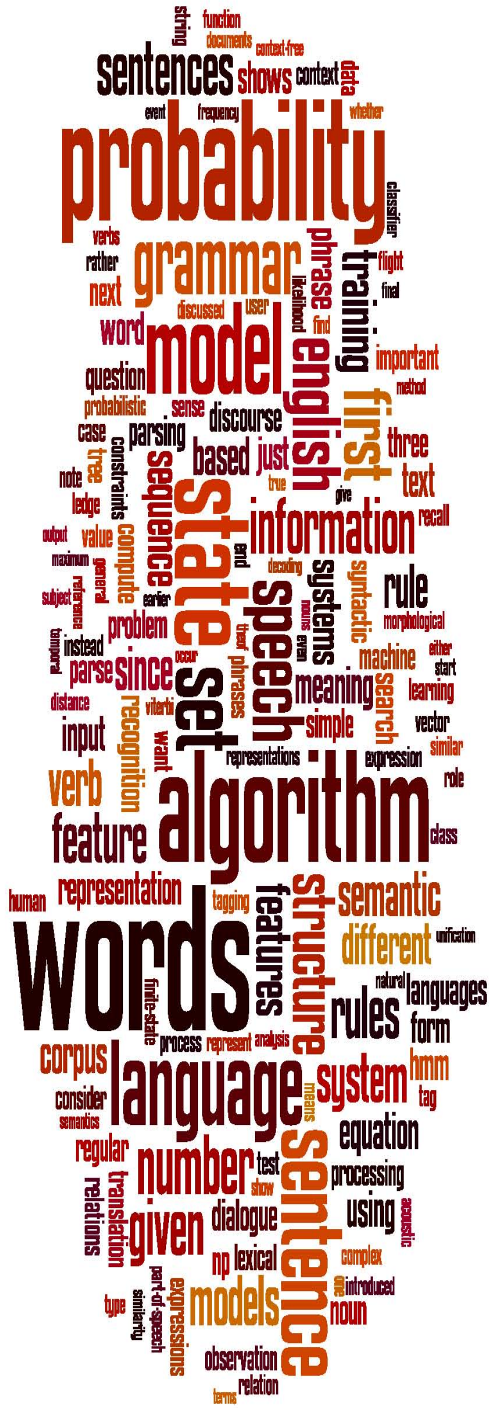


CS 124/LINGUIST 180
From Languages to Information

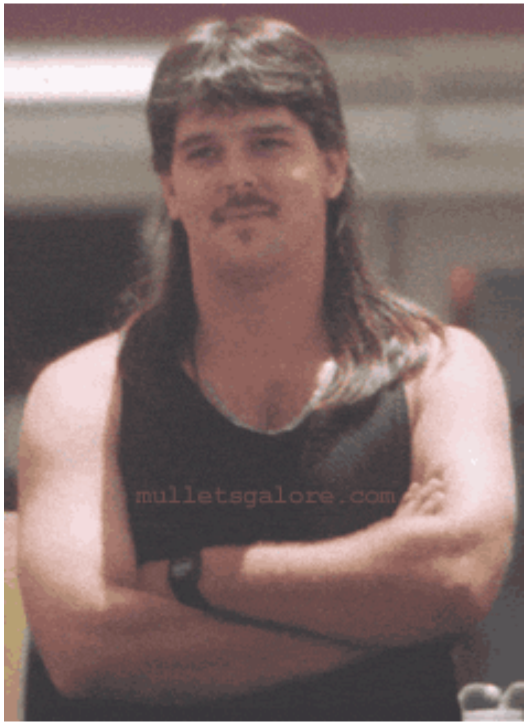
Dan Jurafsky

Stanford University

Recommender Systems & Collaborative Filtering

Slides adapted from Jure Leskovec

Recommender Systems



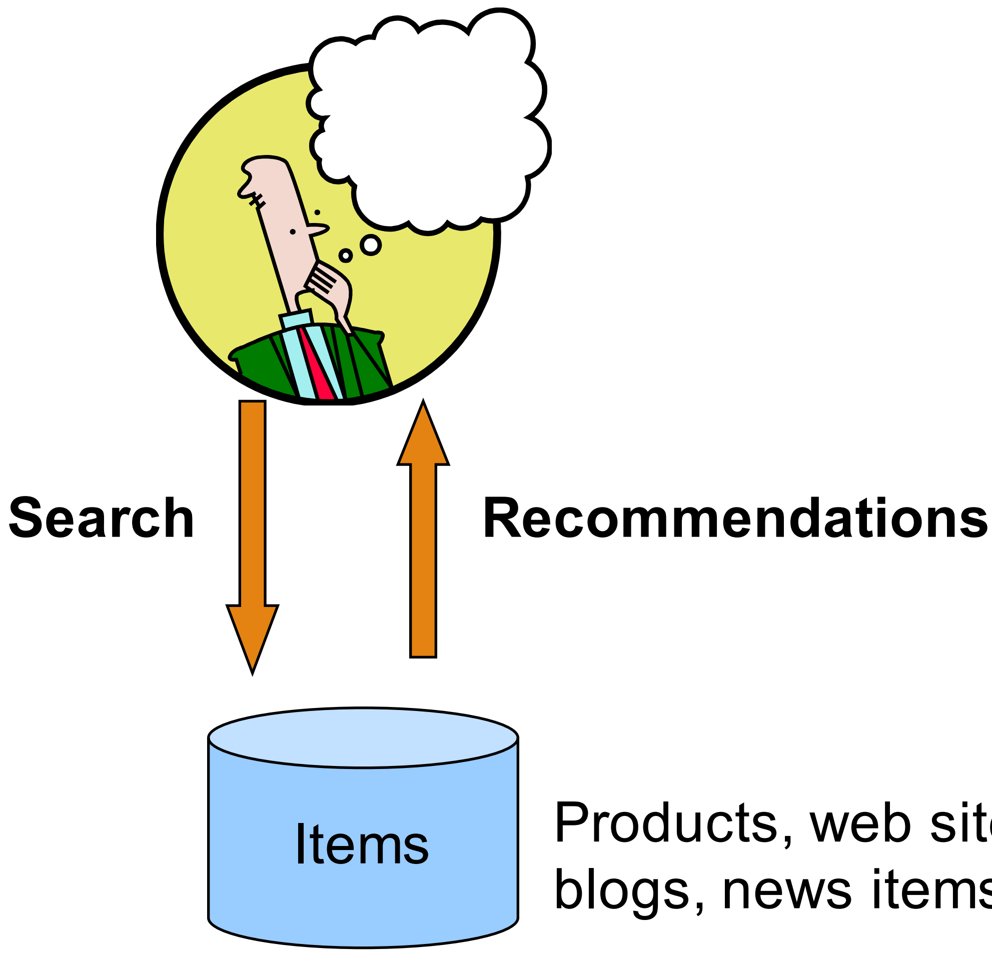
Customer X

- Buys Metallica CD
- Buys Megadeth CD

Customer Y

- Does search on Metallica
- Recommender system suggests Megadeth from data collected about customer **X**

Recommendations



Products, web sites, blogs, news items, …

Examples:

amazon.com
PANDORA
StumbleUpon
NETFLIX
del.icio.us
movielens
helping you find the *right* movies
last.fm
the social music revolution
Google News
YouTube
XBOX LIVE

From Scarcity to Abundance

Shelf space is a scarce commodity for traditional retailers

- Also: TV networks, movie theaters,…

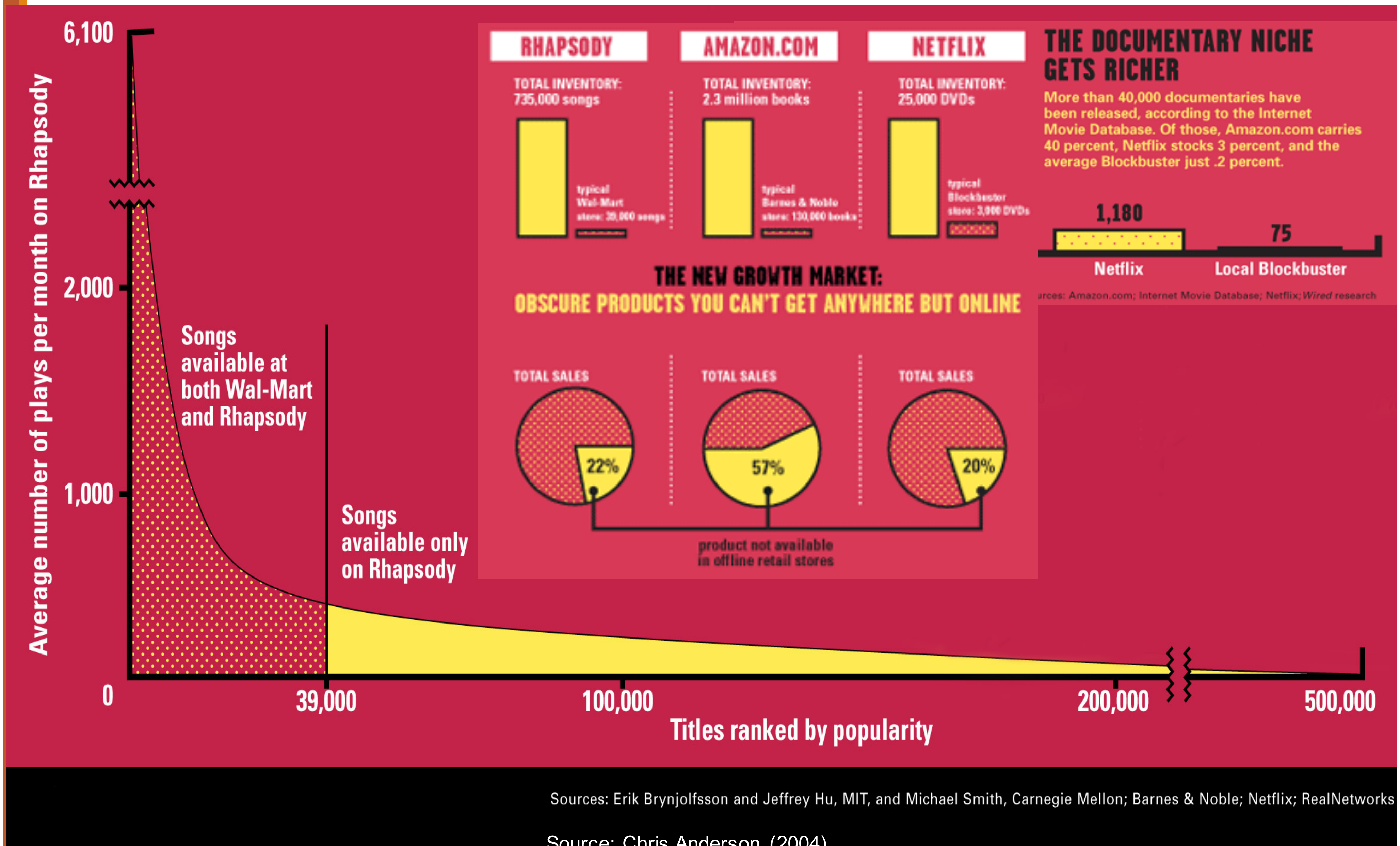
Web enables near-zero-cost dissemination of information about products

- From scarcity to abundance

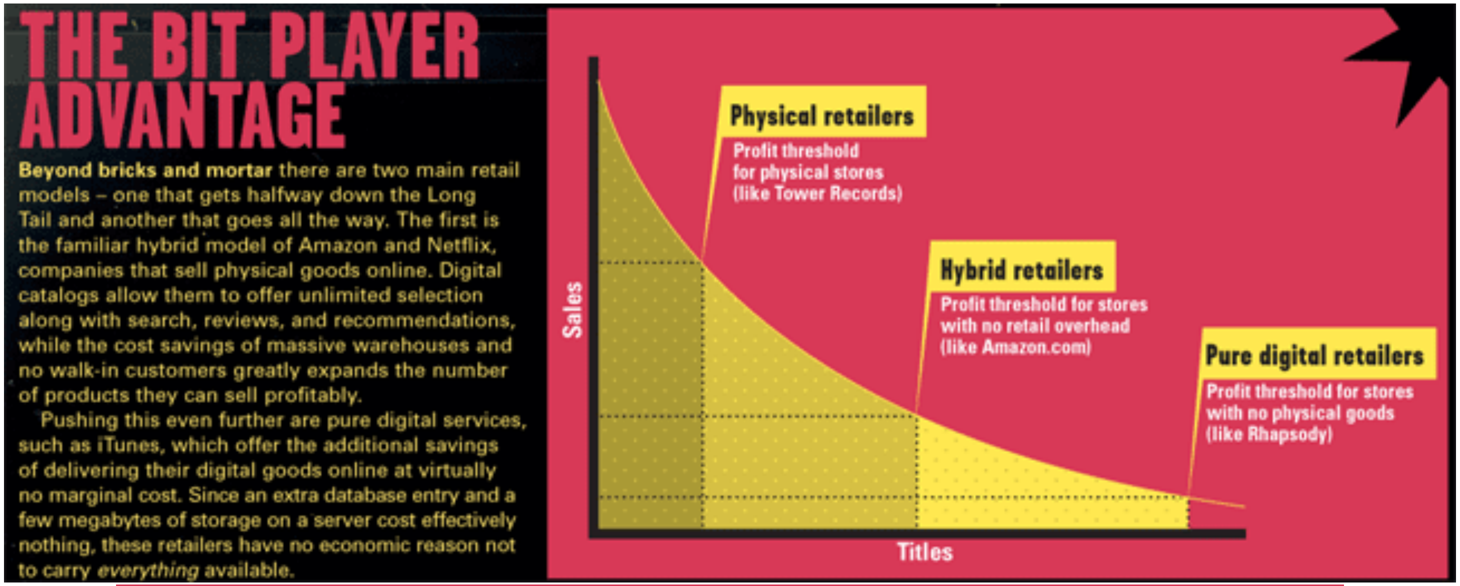
More choice necessitates better filters

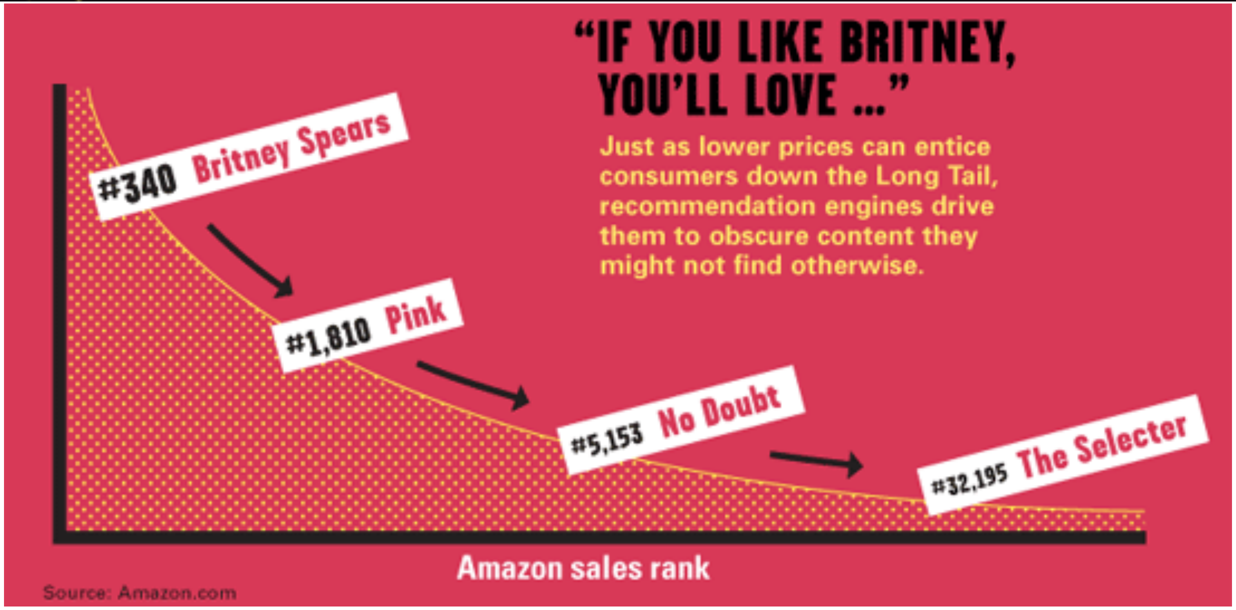
- Recommendation engines
- How **Into Thin Air** made **Touching the Void** a bestseller: http://www.wired.com/wired/archive/12.10/tail.html

Sidenote: The Long Tail



Physical vs. Online





Read http://www.wired.com/wired/archive/12.10/tail.html to learn more!

Types of Recommendations

Editorial and hand curated

- List of favorites
- Lists of "essential" items

Simple aggregates

- Top 10, Most Popular, Recent Uploads

Tailored to individual users

- Amazon, Netflix, …

Today class

Formal Model

$\boldsymbol{X}$ = set of **Customers**

$\boldsymbol{S}$ = set of **Items**

Utility function $\boldsymbol{u}: \boldsymbol{X} \times \boldsymbol{S} \rightarrow \boldsymbol{R}$

- $\boldsymbol{R}$ = set of ratings
- $\boldsymbol{R}$ is a totally ordered set
- e.g., **0-5** stars, real number in **[0,1]**

Utility Matrix

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4

Key Problems

(1) Gathering "known" ratings for matrix

- How to collect the data in the utility matrix

(2) Extrapolate unknown ratings from known ones

- Mainly interested in high unknown ratings
  - We are not interested in knowing what you don't like but what you like

(3) Evaluating extrapolation methods

- How to measure success/performance of recommendation methods

(1) Gathering Ratings

Explicit

- Ask people to rate items
- Doesn't work well in practice – people can't be bothered
- Crowdsourcing: Pay people to label items

Implicit

- Learn ratings from user actions
  - E.g., purchase implies high rating
- What about low ratings?

(2) Extrapolating Utilities

Key problem: Utility matrix ***U*** is **sparse**

- Most people have not rated most items
- **Cold start:**
  - New items have no ratings
  - New users have no history

Three approaches to recommender systems:

1. Content-based
2. Collaborative
3. Latent factor based

Today!

Content-based Recommender Systems

Content-based Recommendations

Main idea: Recommend items to customer ***x*** similar to previous items rated highly by ***x***

Example:

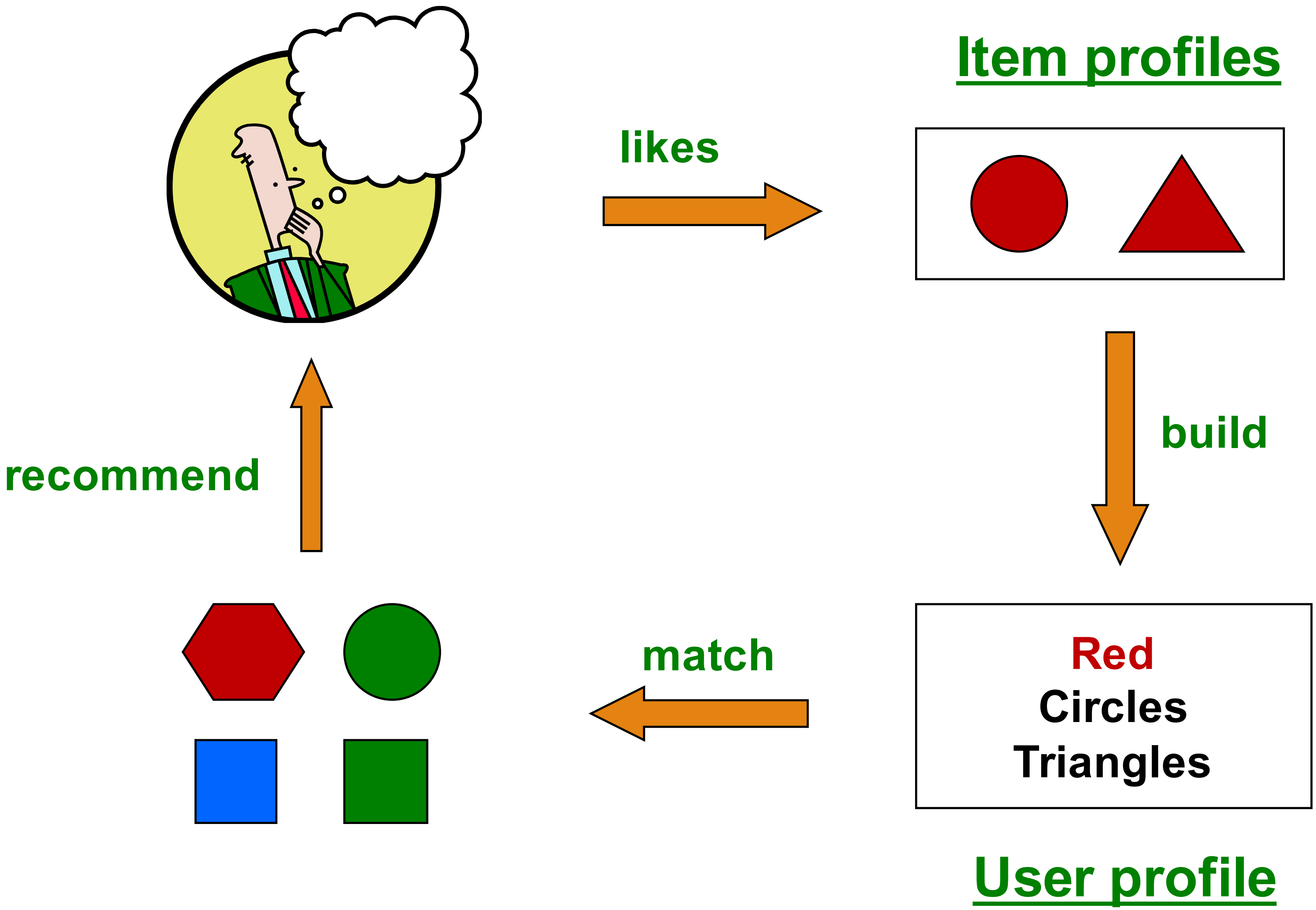
Movie recommendations

- Recommend movies with same actor(s), director, genre, …

Websites, blogs, news

- Recommend other sites with "similar" content

Plan of Action



Item Profiles

For each item, create an **item profile**

Profile is a set (vector) of features

- **Movies:** author, genre, director, actors, year…
- **Text:** Set of "important" words in document

How to pick important features?

- **TF-IDF** (Term frequency * Inverse Doc Frequency)
  - **Term … Feature**
  - **Document … Item**

Content-based Item Profiles

	Melissa McCarthy	Actor A	Actor B	...	Johnny Depp	Comic Genre	Spy Genre	Pirate Genre	Avg Rating
Movie X	0	1	1	0	1	1	0	1	3α
Movie Y	1	1	0	1	0	1	1	0	4α

- Maybe there is a scaling factor α between binary and numeric features
- Or maybe α=1

$$\text{Cosine(Movie X, Movie Y)} = \frac{2+12\alpha^2}{\sqrt{5+9\alpha^2}\sqrt{5+16\alpha^2}}$$

User Profiles

Want a vector with the same components/dimensions as items

- Could be 1s representing user purchases
- Or arbitrary numbers from a rating

User profile is aggregate of items:

- Average(weighted?)of rated item profiles

Sample user profile

- Items are movies
- Utility matrix has 1 if user has seen movie
- 20% of the movies user U has seen have Melissa McCarthy
- U["Melissa McCarthy"] = 0.2

	Melissa McCarthy	Actor A	Actor B	...	
User U	0.2	.005	0	0	...

Prediction

- User and item vectors have the same components/dimensions!
- So just recommend the items whose vectors are most similar to the user vector!

- Given user profile $\boldsymbol{x}$ and item profile $\boldsymbol{i}$,
- estimate $u(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$

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

- No first-rater problem

+: Able to provide explanations

- Can provide explanations of recommended items by listing content-features that caused an item to be recommended

Cons: Content-based Approach

–: Finding the appropriate features is hard

- E.g., images, movies, music

–: Recommendations for new users

- **How to build a user profile?**

–: Overspecialization

- Never recommends items outside user’s content profile
- People might have multiple interests
- **Unable to exploit quality judgments of other users**

Collaborative Filtering

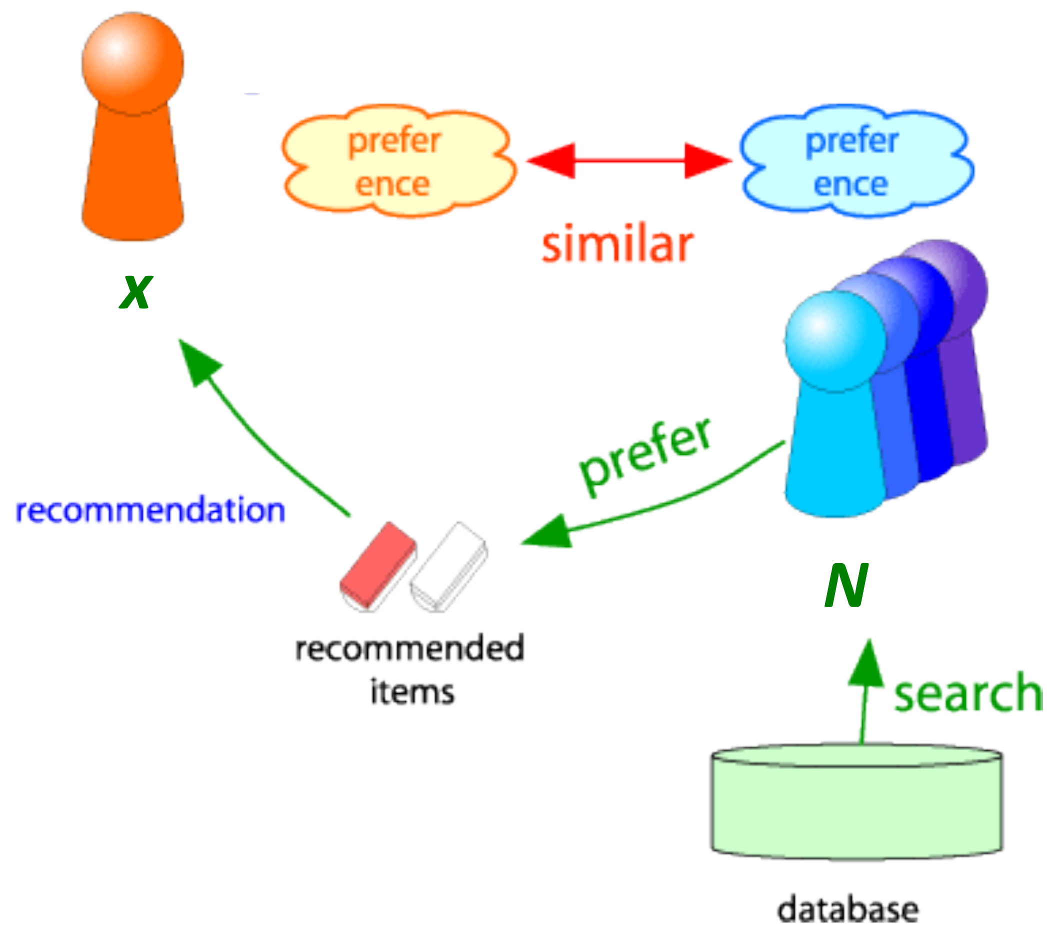
Harnessing quality judgments of other users

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

$\boldsymbol{r_x}$ = [*, _, _, *, ***]
$\boldsymbol{r_y}$ = [*, _, **, **, _]

Let $\boldsymbol{r_x}$ be the vector of user $\boldsymbol{x}$'s ratings

Jaccard similarity measure

- **Problem:** Ignores the value of the rating

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

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"

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

Utility Matrix

	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: 1/5 < 2/4

Cosine similarity: 0.386 > 0.322

Considers missing ratings as "negative"

Utility Matrix

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

- Problem with cosine: 0 acts like a negative review
  - C really loves SW
  - A hates SW
  - B just hasn't seen it
- Another problem: we'd like to normalize for raters
  - D rated everything the same; not very useful

Modified Utility Matrix: subtract the means of each row

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

	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

- Now a 0 means no information
- And negative ratings means viewers with opposite ratings will have vectors in opposite directions!

Modified Utility Matrix: subtract the means of each row

	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

Cos(A,B) = $$\frac{(2/3)\times(1/3)}{\sqrt{(2/3)^2+(5/3)^2+(-7/3)^2}\sqrt{(1/3)^2+(1/3)^2+(-2/3)^2}} = 0.092$$

Cos(A,C) = $$\frac{(5/3)\times(-5/3)+(-7/3)\times(1/3)}{\sqrt{(2/3)^2+(5/3)^2+(-7/3)^2}\sqrt{(-5/3)^2+(1/3)^2+(4/3)^2}} = -0.559$$

Now A and C are (correctly) way further apart than A,B

Cosine after subtracting mean

Turns out to be the same as Pearson correlation coefficient!!!

Cosine similarity is correlation when the data is centered at 0

- Terminological Note: subtracting the mean is **zero-centering**, not **normalizing** (normalizing is dividing by a norm to turn something into a probability), but the textbook (and in common use) we sometimes overload the term "normalize"

Finding Similar Users

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

Let r_x be the vector of user x's ratings

Cosine similarity measure

- sim(x, y) = cos(r_x, r_y) = $\frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$
- **Problem:** Treats missing ratings as "negative"

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

Pearson correlation coefficient

- S_{xy} = items rated by both users x and y

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

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

Rating Predictions

From similarity metric to recommendations:

Let $\boldsymbol{r_x}$ be the vector of user $\boldsymbol{x}$'s ratings

Let $\boldsymbol{N}$ be the set of $\boldsymbol{k}$ users most similar to $\boldsymbol{x}$ who have rated item $\boldsymbol{i}$

Prediction for item *i* of user *x*:

- $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
- $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$

- **Many other tricks possible…**

Shorthand:
$\boldsymbol{s_{xy} = sim(x, y)}$

Item-Item Collaborative Filtering

So far: User-user collaborative filtering

Another view: Item-item

- For item ***i***, find other similar items
- Estimate rating for item ***i*** based on ratings for similar items
- Can use same similarity metrics and prediction functions as in user-user model

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

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

Item-Item CF (|N|=2)

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-Item CF (|N|=2)

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-Item CF (|N|=2)

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		**0.59**

Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Here we use Pearson correlation as similarity:

1) Subtract mean rating m_i from each movie i

$m_1 = (1+3+5+5+4)/5 = \mathbf{3.6}$

row 1: *[-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]*

2) Compute cosine similarities between rows

Item-Item CF (|N|=2)

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		0.41
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		0.59

Compute similarity weights:

$s_{1,3}=0.41$, $s_{1,6}=0.59$

Item-Item CF (|N|=2)

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		**2.6**	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	

Predict by taking weighted average:

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

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

Item-Item vs. User-User

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.8	
Bob		0.5		0.3
Carol	0.9		1	0.8
David			1	0.4

- **In practice, it has been observed that <u>item-item</u> often works better than user-user**
- **Why?** Items are simpler, users have multiple tastes

Pros/Cons of Collaborative Filtering

+ Works for any kind of item

- No feature selection needed

- Cold Start:

- Need enough users in the system to find a match

- Sparsity:

- The user/ratings matrix is sparse
- Hard to find users that have rated the same items

- First rater:

- Cannot recommend an item that has not been previously rated
- New items, Esoteric items

- Popularity bias:

- Cannot recommend items to someone with unique taste
- Tends to recommend popular items

Hybrid Methods

Implement two or more different recommenders and combine predictions

- Perhaps using a linear model

Add content-based methods to collaborative filtering

- Item profiles for new item problem
- Demographics to deal with new user problem

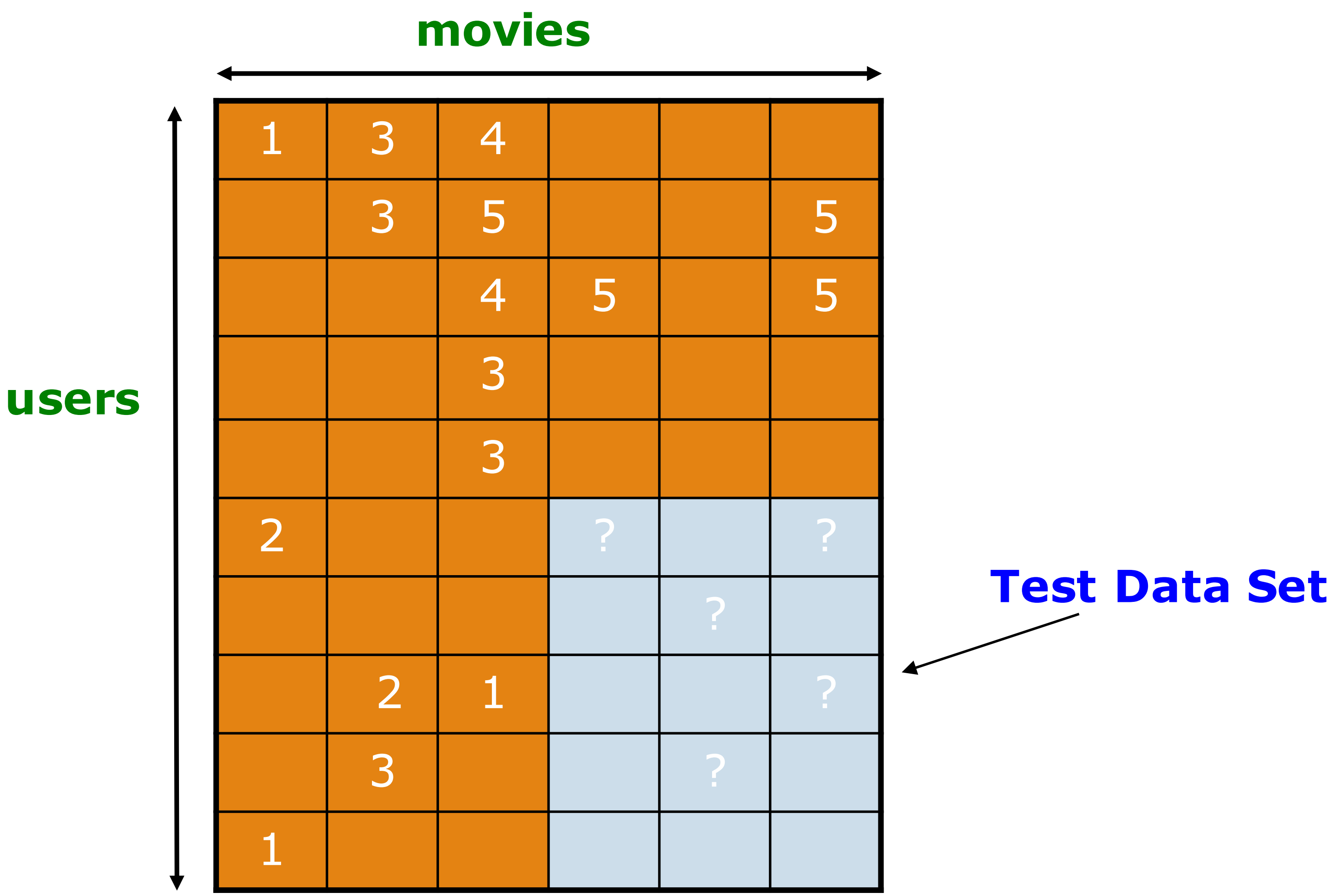
Evaluation

movies

users

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

Evaluation



Evaluating Predictions

Compare predictions with known ratings

- **Root-mean-square error** (RMSE)
  - $\sqrt{\sum_{xi}(r_{xi} - r^*_{xi})^2}$ where $\boldsymbol{r_{xi}}$ is predicted, $\boldsymbol{r^*_{xi}}$ is the true rating of ***x*** on ***i***
- **Rank Correlation**:
  - Spearman's *correlation* between system's and user's complete rankings

Problems with Error Measures

Narrow focus on accuracy sometimes misses the point

- Prediction Diversity
- Prediction Context
- Order of predictions

In practice, we care only to predict high ratings:

- RMSE might penalize a method that does well for high ratings and badly for others

There's No Data like Mo' Data

Leverage all the data

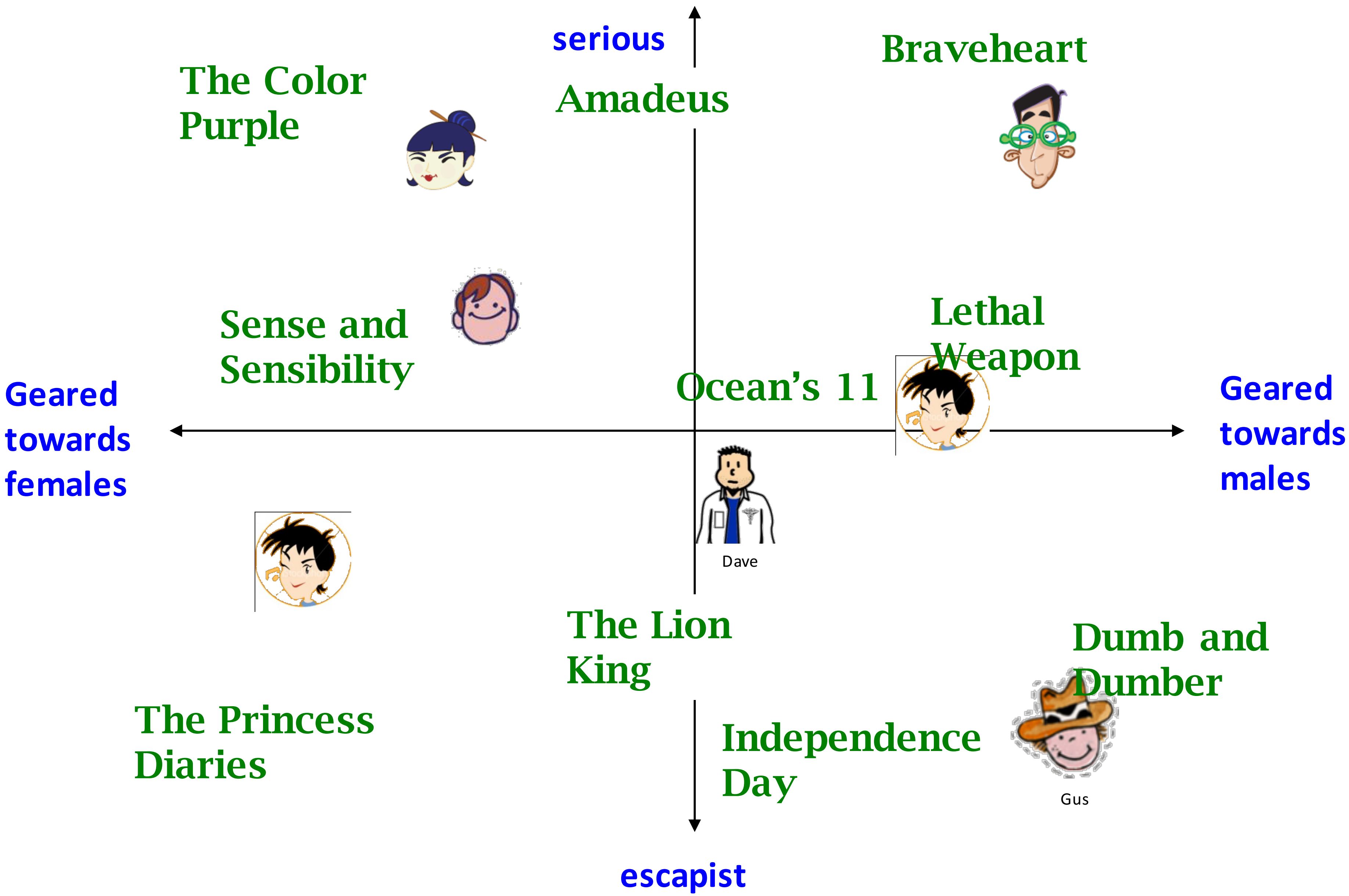
- Simple methods on large data do best

Add more data

- e.g., add IMDB data on genres

More data beats better algorithms

Latent Factor Models (like SVD)



Famous Historical Example: The Netflix Prize

Training data
- 100 million ratings, 480,000 users, 17,770 movies
- 6 years of data: 2000-2005

Test data
- Last few ratings of each user (2.8 million)
- Evaluation criterion: root mean squared error (RMSE)
- Netflix Cinematch RMSE: 0.9514

Competition
- 2700+ teams
- $1 million prize for 10% improvement on Cinematch
- BellKor system won in 2009. Combined many factors
  - Overall deviations of users/movies
  - Regional effects
  - Local collaborative filtering patterns
  - Temporal biases

Summary on Recommendation Systems

- The Long Tail
- Content-based Systems
- Collaborative Filtering
- Latent Factors