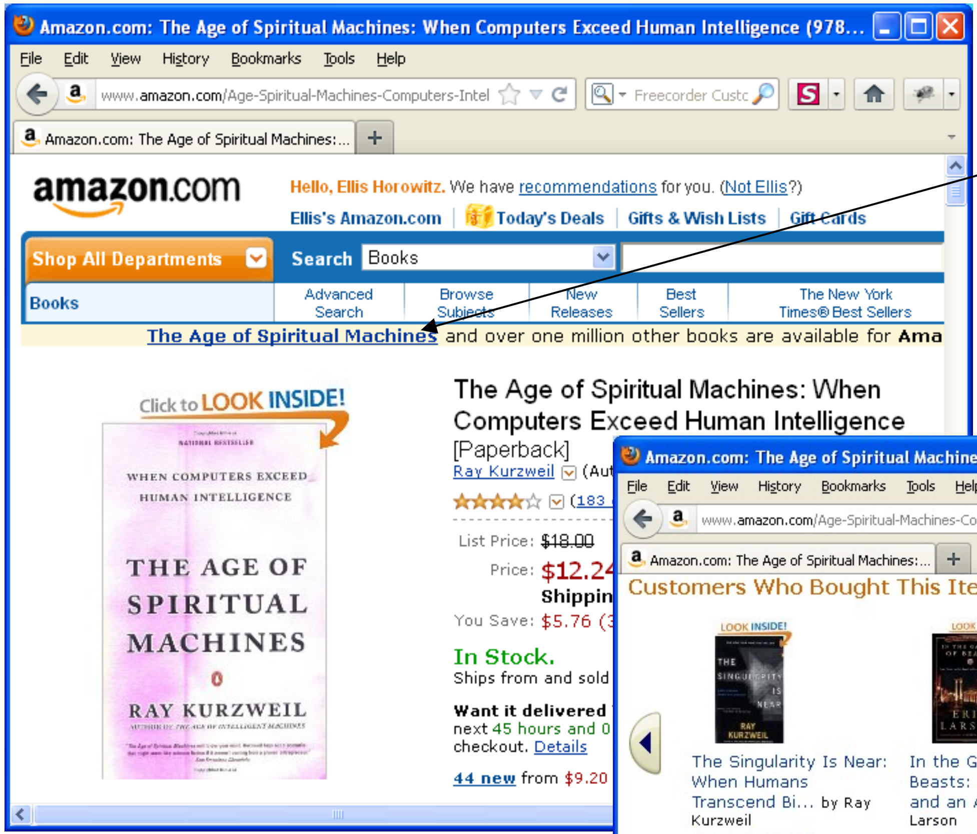
Recommendation Systems

Collaborative Filtering & Content-Based Recommendations

USC Viterbi School of Engineering

Recommendations are Found Everywhere

Amazon uses recommendations to great effect; here a user looks at a book, *Age of Spiritual Machines* and Amazon's recommendations


Amazon.com: The Age of Spiritual Machines: When Computers Exceed Human Intelligence (978...

www.amazon.com/Age-Spiritual-Machines-Computers-Intel

Hello, Ellis Horowitz. We have recommendations for you. (Not Ellis?)

The Age of Spiritual Machines and over one million other books are available for Ama

The Age of Spiritual Machines: When Computers Exceed Human Intelligence [Paperback]

Ray Kurzweil (Au

List Price: ~~$18.00~~

Price: $12.24

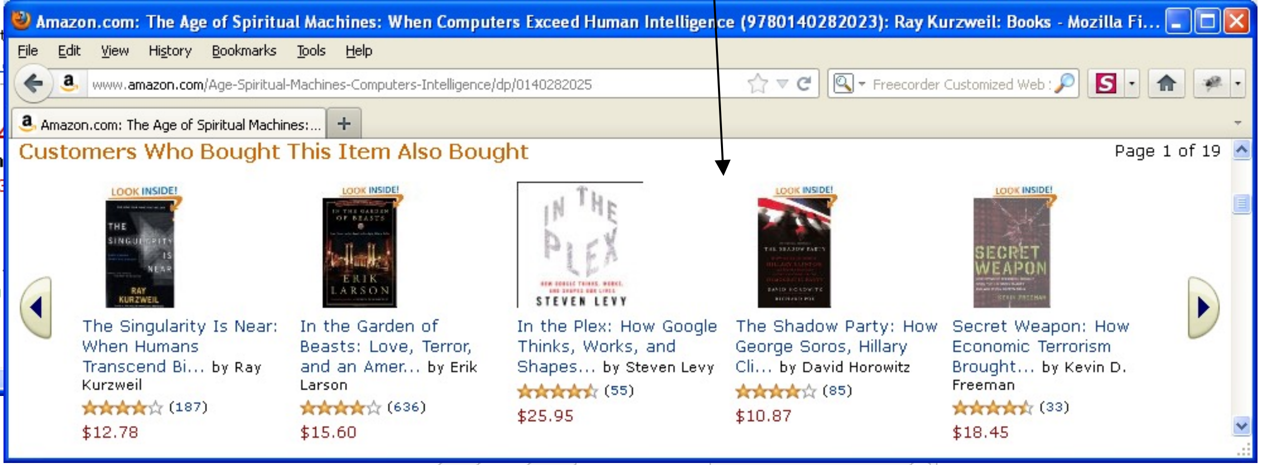
You Save: $5.76

In Stock.

Ships from and sold

Want it delivered next 45 hours and 0 checkout. Details

44 new from $9.20


Amazon.com: The Age of Spiritual Machines: When Computers Exceed Human Intelligence (9780140282023): Ray Kurzweil: Books - Mozilla Fi...

www.amazon.com/Age-Spiritual-Machines-Computers-Intelligence/dp/0140282025

Customers Who Bought This Item Also Bought

Page 1 of 19

The Singularity Is Near: When Humans Transcend Bi... by Ray Kurzweil	In the Garden of Beasts: Love, Terror, and an Amer... by Erik Larson	In the Plex: How Google Thinks, Works, and Shapes... by Steven Levy	The Shadow Party: How George Soros, Hillary Cli... by David Horowitz	Secret Weapon: How Economic Terrorism Brought... by Kevin D. Freeman
(187)	(636)	(55)	(85)	(33)
$12.78	$15.60	$25.95	$10.87	$18.45

Value of Recommendation Systems

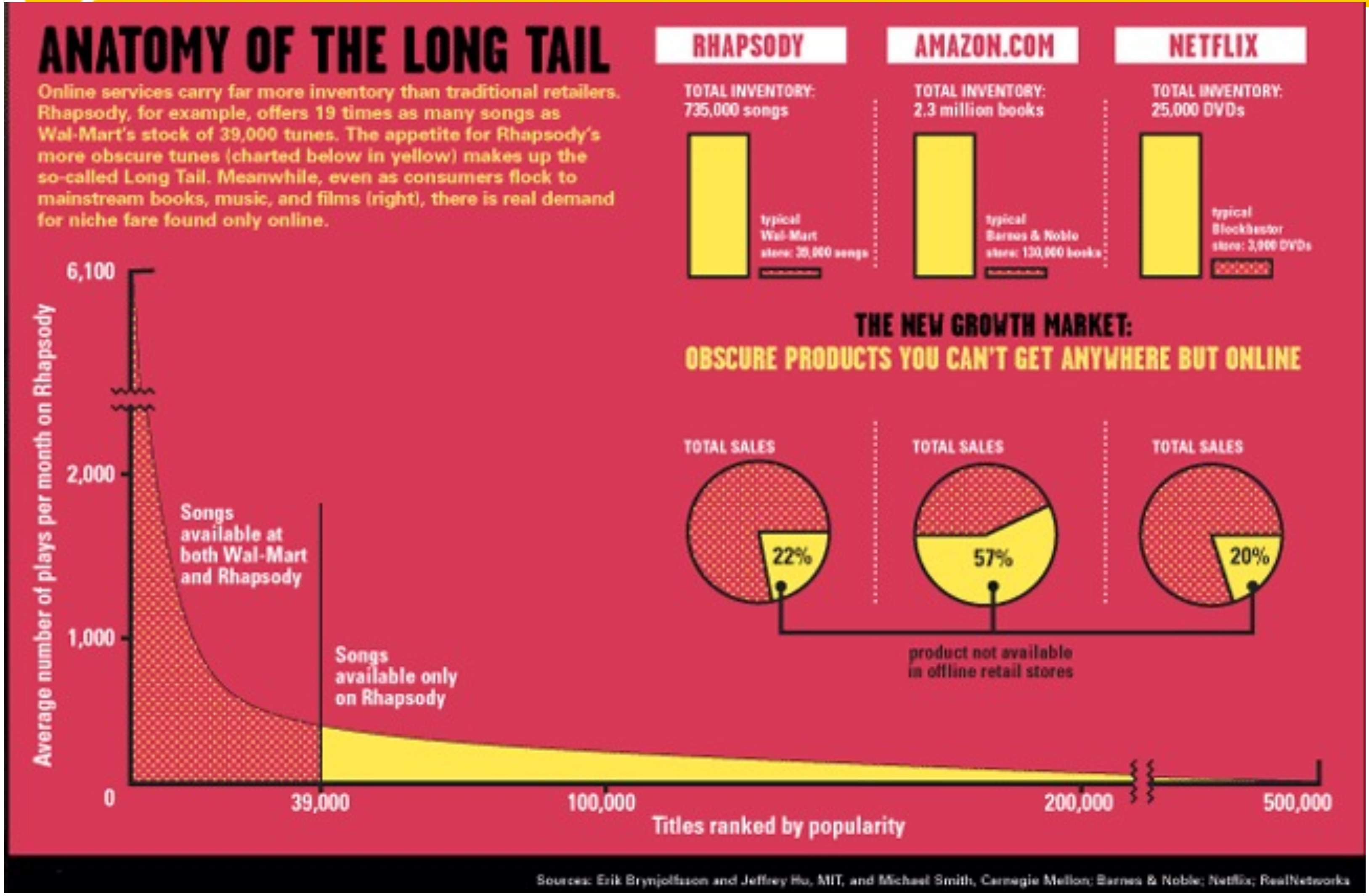
- Recommendation Systems have been shown to substantially increase sales at on-line stores
  - Recommendation engines can significantly increase revenue, improve CTRs and conversions. It also contributes to the improvement of factors more difficult to measure, such as customer satisfaction, and it increases customer retention.
  - https://neoteric.eu/blog/how-to-boost-sales-with-a-recommender-system/
  - "YouTube's video recommending algorithm is, literally, worth billions of dollars:" the company's CPO, Neal Mohan, reports that over 70 percent of the time spent on YouTube is spent watching recommended videos
  - https://faun.pub/the-algorithm-worth-billions-how-youtubes-addictive-video-recommender-works-d75646dac6a3

The YouTube Recommendation Algorithm Why All the Fuss

- **“YouTube’s recommender AI still a horror show, finds major crowdsourced study”**
- **https://techcrunch.com/2021/07/07/youtubes-recommender-ai-still-a-horrorshow-finds-major-crowdsourced-study/**
- New research published July 2021 by Mozilla backs that notion up, suggesting YouTube’s AI continues to puff up piles of “bottom-feeding”/low-grade/divisive/disinforming content — stuff that tries to grab eyeballs by triggering people’s sense of outrage, sewing division/polarization or spreading baseless/harmful disinformation — which in turn implies that YouTube’s problem with recommending terrible stuff is indeed systemic; a side effect of the platform’s rapacious appetite to harvest views to serve ads.

USC Viterbi School of Engineering

Why Recommendation Systems Are Needed



ANATOMY OF THE LONG TAIL

Online services carry far more inventory than traditional retailers. Rhapsody, for example, offers 19 times as many songs as Wal-Mart's stock of 39,000 tunes. The appetite for Rhapsody's more obscure tunes (charted below in yellow) makes up the so-called Long Tail. Meanwhile, even as consumers flock to mainstream books, music, and films (right), there is real demand for niche fare found only online.

THE NEW GROWTH MARKET: OBSCURE PRODUCTS YOU CAN'T GET ANYWHERE BUT ONLINE

RHAPSODY
TOTAL INVENTORY: 735,000 songs
typical Wal-Mart store: 39,000 songs
AMAZON.COM
TOTAL INVENTORY: 2.3 million books
typical Barnes & Noble store: 130,000 books
NETFLIX
TOTAL INVENTORY: 25,000 DVDs
typical Blockbuster store: 3,000 DVDs
TOTAL SALES
22%
TOTAL SALES
57%
TOTAL SALES
20%
product not available in offline retail stores
Average number of plays per month on Rhapsody
6,100
2,000
1,000
0
Songs available at both Wal-Mart and Rhapsody
Songs available only on Rhapsody
39,000
100,000
200,000
500,000
Titles ranked by popularity

Sources: Erik Brynjolfsson and Jeffrey Hu, MIT, and Michael Smith, Carnegie Mellon; Barnes & Noble; Netflix; RealNetworks

http://www.wired.com/wired/archive/12.10/tail.html?pg=3

Scarcity versus Abundance

- **Since online systems maintain large quantities of goods, systems that provide recommendations serve an important purpose**
  - In some cases items sold from the long tail, (i.e. those not particularly popular) can cumulatively outweigh the initial portion of the graph, an in effect produce the majority of sales
- **Recommendation systems are expected to increase diversity because they help us discover new products.**
  - However, some algorithms may unintentionally do the opposite.
  - Because they recommend products based on past sales or ratings, they cannot usually recommend products with limited historical data. This can create a rich-get-richer effect for popular products
  - This bias toward popularity can prevent what are otherwise better consumer-product matches.
    - Several collaborative filtering algorithms have been developed to promote diversity and the long tail by recommending novel, unexpected, and serendipitous items
  - See https://en.wikipedia.org/wiki/Collaborative_filtering

USC Viterbi School of Engineering

Two Types of Recommendation Systems

- A *recommendation system* is any system which provides a recommendation/prediction/opinion to a user on items

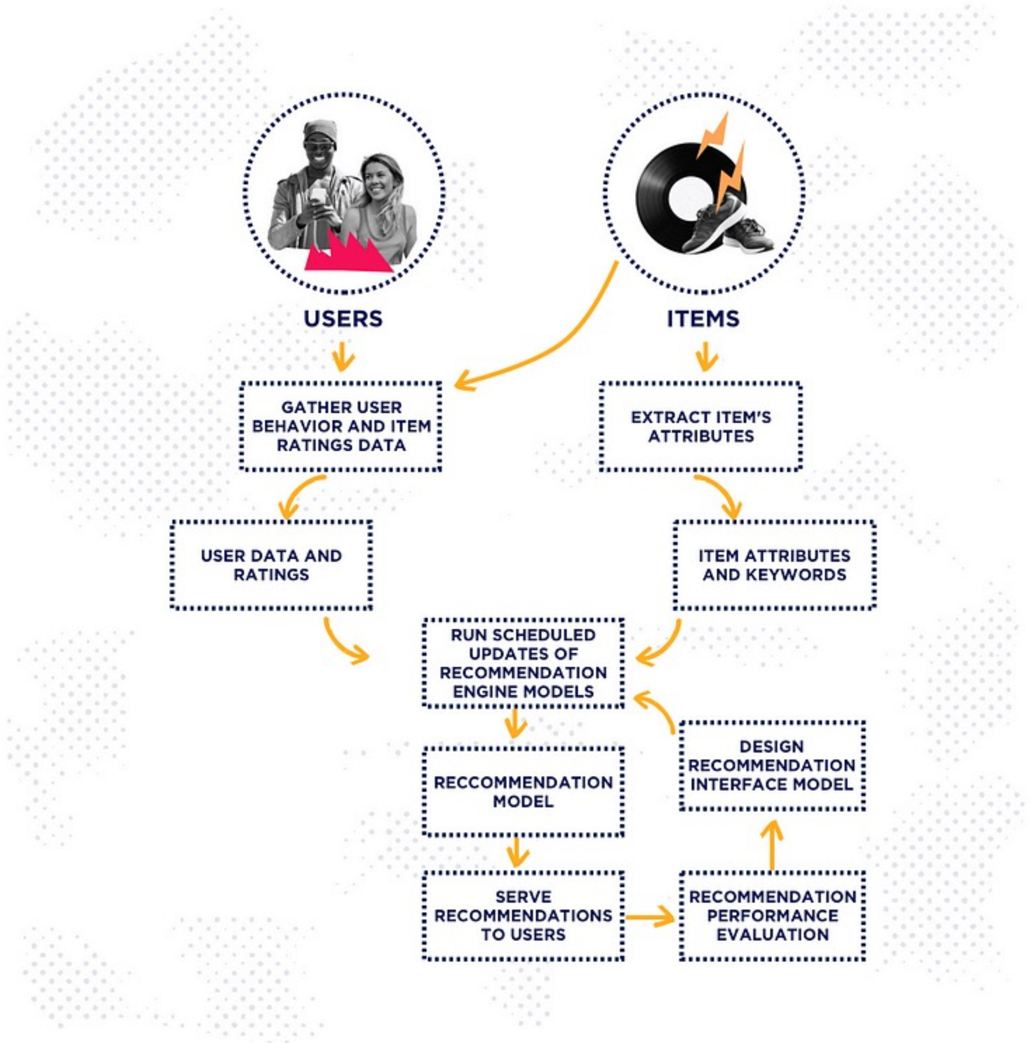
1. A classic *content-based filtering system* uses item similarity/clustering to recommend items like ones you like
2. A classic *collaborative filtering system* uses user similarity, i.e. the links between users and the item they chose as the basis of recommendations

- Commonly many companies use *hybrid systems* in which both kinds of techniques are employed

Difference between Collaborative and Content-based Filtering-An Example

- *Here are two early systems that recommended music*
- *Last.fm* creates a "station" of recommended songs by observing what bands and individual tracks ***the user*** has listened to on a regular basis (*user similarity*) and comparing those against the listening behavior of other users.
  - Last.fm will play tracks that do not appear in the user's library, but are often played by other users with similar interests.
  - As this approach leverages the behavior of users, it is an example of a ***collaborative filtering technique***.
- *Pandora* uses the ***properties of a song*** or artist (a subset of the 400 attributes provided by the Music Genome Project) to seed a "station" that plays music with similar properties (*item similarity*).
  - User feedback is used to refine the station's results, deemphasizing certain attributes when a user "dislikes" a particular song and emphasizing other attributes when a user "likes" a song.
  - This is an example of a ***content-based technique***.

Basic Flowchart of a Recommender System


USERS
ITEMS
GATHER USER BEHAVIOR AND ITEM RATINGS DATA
EXTRACT ITEM'S ATTRIBUTES
USER DATA AND RATINGS
ITEM ATTRIBUTES AND KEYWORDS
RUN SCHEDULED UPDATES OF RECOMMENDATION ENGINE MODELS
DESIGN RECOMMENDATION INTERFACE MODEL
RECCOMMENDATION MODEL
SERVE RECOMMENDATIONS TO USERS
RECOMMENDATION PERFORMANCE EVALUATION

An Example Restaurant Recommendations

- Suppose we have a list of all Los Angeles restaurants
  - with ↑ and ↓ ratings for *some*
  - as provided by USC students
- Which restaurant(s) should I recommend to you?

Input

Alice	Il Fornaio	Yes
Bob	Ming's	No
Cindy	Straits Café	No
Dave	Ming's	Yes
Alice	Straits Café	No
Estie	Zao	Yes
Cindy	Zao	No
Dave	Brahma Bull	No
Dave	Zao	Yes
Estie	Ming's	Yes
Fred	Brahma Bull	No
Alice	Mango Café	No
Fred	Ramona's	No
Dave	Homma's	Yes
Bob	Higashi West	Yes
Estie	Straits Café	Yes

Algorithm 0

- Strategy: Recommend to you the most popular restaurants
  - say # positive votes minus # negative votes
- But this ignores
  - your culinary preferences
  - judgments of those with similar preferences
- How can we exploit the wisdom of "like-minded" people?
- Basic assumption
  - Preferences are not random
  - Assumption: if I like Il Fornaio, it's more likely I will also like Cenzo (another Italian restaurant)

Cast the Input as a Matrix

	Brahma Bull	Higashi West	Mango	Il Fornaio	Zao	Ming's	Ramona's	Straits	Homma's
Alice		Yes	No	Yes				No	
Bob		Yes				No		No	
Cindy				Yes	No			No	
Dave	No			No	Yes	Yes			Yes
Estie				No	Yes	Yes		Yes	
Fred	No						No		

Called a *utility matrix*

Each row represents an individual and each column a restaurant
In this example, entries are either yes/no;
In the more general case they can be any value

USC Viterbi School of Engineering

The Utility Matrix

- In a recommendation-system application there are two classes of entities, which we shall refer to as *users* and *items*
  - Users have preferences for certain items, and these preferences must be teased out of the data.
- The data itself is represented as a ***utility matrix***, giving for each user-item pair, a value that represents what is known about the degree of preference of that user for that item.
- *Values might come from an ordered set*, e.g., integers 1–5 representing the number of stars that the user gave as a rating for that item.
- We assume that the matrix is *sparse*, meaning that most entries are “unknown.”
- An unknown rating implies that we have no explicit information about the user’s preference for the item.

A New Utility Matrix with Rankings

	Brahma Bull	Higashi West	Mango	Il Fornaio	Zao	Ming's	Ramona's	Straits	Homma's
Alice		1	-1	1				-1	
Bob		1				-1		-1	
Cindy				1	-1			-1	
Dave	-1			-1	1	1			1
Estie				-1	1	1		1	
Fred	-1						-1		

View all other entries as zeros for now.

- To compute the similarity between individual's preference vectors we can use inner products as a good place to start, e.g.
    - Dave has similarity 3 with Estie,
        - e.g. (-1,0,0,-1,1,1,0,0,1) and (0,0,0,-1,1,1,0,1,0)
        - (i.e. there are three matching values of either 1 or -1)
    - but -2 with Cindy (0,0,0,1,-1,0,0,-1,0) (a zero value doesn't count).
- Perhaps recommend Straits Cafe to Dave and Il Fornaio to Bob, etc.

Another Utility Matrix Example - Movies

	Avatar	LOTR	MATRIX	PIRATES
ALICE	1		0.2	
BOB		0.5		0.3
CAROL	0.2		1	
DAVID				0.4

- The blank spaces indicate either the user has not seen the movie or has decided not to rate it
- Main issue: how to fill in the values in the missing fields
- In general there are two basic techniques for populating the utility matrix
  1. Ask users to rate items
     E.g. movies, online stores from purchasers
  2. Make inferences from user behaviors
     Assumption: Users who watch a movie must “like” it, so above we can assign it a 1;
     We are mostly interested in fields for which the ratings are high

Recommending Documents Also can be Viewed as a Form of Recommendation System

- **If the items we are considering are documents, then the profile will be the set of "important" words in the document**
- **How do we pick important words**
  - **We use the TF-IDF formulation seen earlier**

Profile of a document d_j is the vector of weights $w_{i,j}$ $Content(d_j) = (w_{1j}, \ldots w_{kj}).$

$$w_{i,j} = TF_{i,j} \times IDF_i \qquad TF_{i,j} = \frac{f_{i,j}}{\max_z f_{z,j}}, \qquad IDF_i = \log \frac{N}{n_i}.$$

- ***TF*** **: Term Frequency, IDF : Inverse Document Frequency**
- ***N*** **: Number of the documents**
- **n_i : How many times an element is seen in all of the documents**
- **$f_{i,j}$: Number of times an element is seen in the document d_j**

Example 1 Boolean Utility Matrix

- **Items are movies, only feature is Actor**
  - **Item profile: vector with *0* or *1* for each actor**
- **Suppose user *X* has watched *5* movies**
  - ***2* movies featuring actor A (movies 1 and 3)**
  - ***3* movies featuring actor B (movies 2, 4, and 5)**
- **User profile = mean of item profiles**
  - **Feature A's weight = *2/5* = *0.4***
  - **Feature B's weight = *3/5* = *0.6***

	(ActA,	ActB,	ActC,	ActD,	ActE)
Movie1	(1,	0,	0,	0,	0)
Movie2	(0,	1,	0,	0,	0,)
Movie3	(1,	0,	0,	0,	0)
Movie4	(0,	1,	0,	0,	0)
Movie5	(0,	1,	0,	0,	0)

ActA's weight = Sum(ActA)/5

Example 2 Star Ratings Normalized

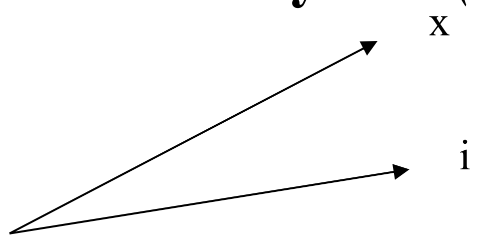
- Same example, 1-5 star ratings
  - Actor A's movies rated *3* and *5*
  - Actor B's movies rated *1, 2,* and *4* (note: *1* and *2* are negative ratings)
- Useful step: normalize ratings by subtracting user's mean rating (3)
  - Given ratings 1,2,3,4,5 – the mean is 3, so:
  - Actor A's normalized ratings = 0, +2
    - Profile weight = (0 + 2)/2 = 1
  - Actor B's normalized ratings = -2, -1, +1
    - Profile weight = -2/3

	ActA	ActB	ActC	ActD	ActE
Movie1	3	0	0	0	0
Movie2	0	1	0	0	0
Movie3	5	0	0	0	0
Movie4	0	2	0	0	0
Movie5	0	4	0	0	0

	ActA	ActB	ActC	ActD	ActE
Movie1	0	0	0	0	0
Movie2	0	-2	0	0	0
Movie3	2	0	0	0	0
Movie4	0	-1	0	0	0
Movie5	0	1	0	0	0

Making Predictions

- **Given user profile *x* (movies he/she watched) and item profile *i (movies with actor profiles)***
- **Estimate the similarity of $U(x,i) = \cos(\theta) = (x.i)/(|x|\ |i|)$**



- **Technically the cosine distance is actually the angle θ and the cosine similarity is the angle $180 - \theta$**

- **For convenience we use $\cos(\theta)$ as our similarity measure and call it the "cosine similarity" in this context**

Pros: Content-based Approach

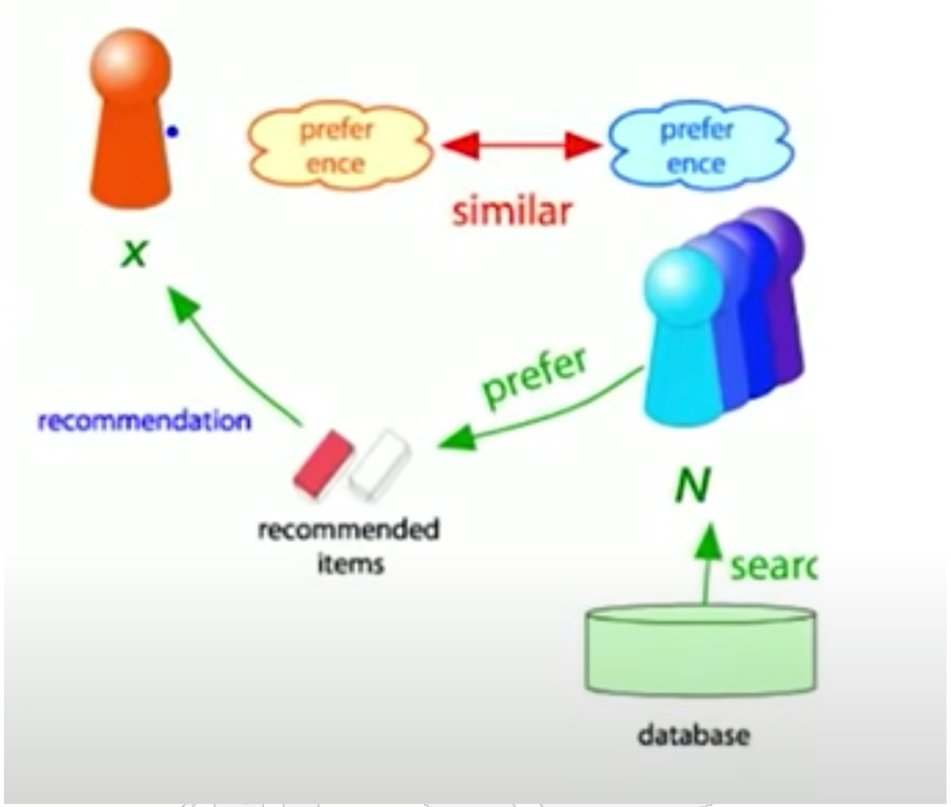
- No need for data on other users
- Able to recommend to users with unique tastes
- Able to recommend new and unpopular items
  - No first rater problem
- Explanations for recommended items
  - Content features that caused an item to be recommended

Cons: Content-Based Approach

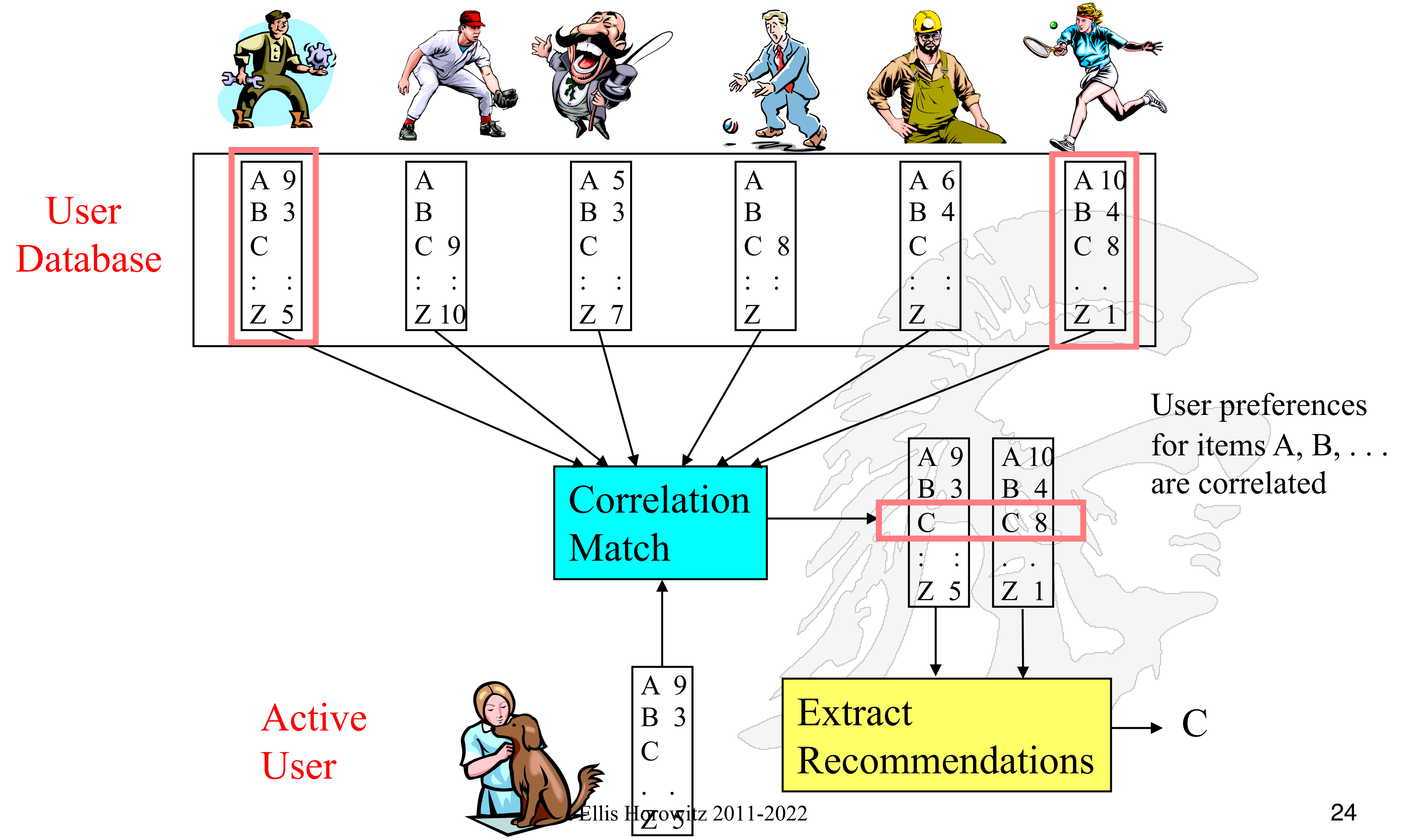
- Finding the appropriate features is not always obvious
  - E.g. movie features may be obvious but what about the features for images and music
- Overspecialization
  - Never recommends items outside user's content profile
  - People might have multiple interests
  - Unable to exploit quality judgements of other users
- Cold-start problem for new users
  - How to build a user profile

Let’s Switch Focus to Collaborative Filtering

- Collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many other users (collaborating).
  - The underlying assumption of the collaborative filtering approach is that if a person *A* has the same opinion as a person *B* on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person



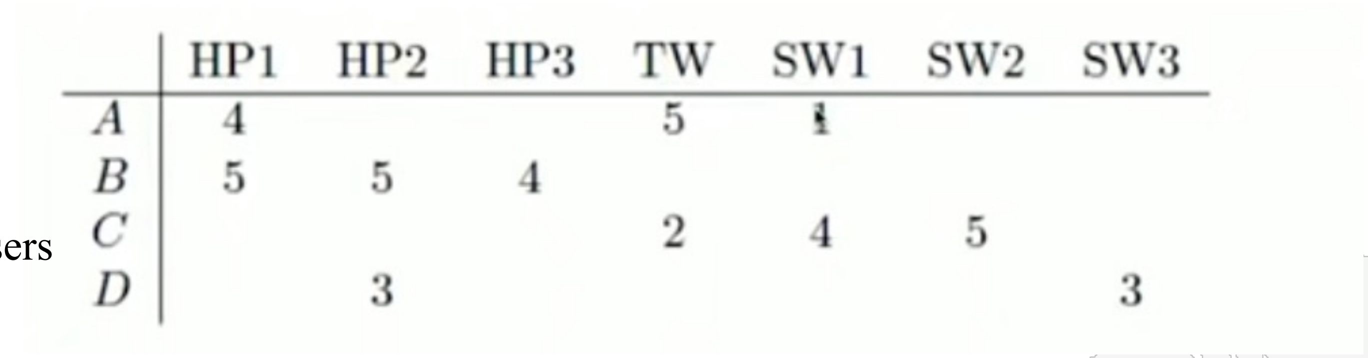
Collaborative Filtering



Similar Users and Jaccard Similarity

movies

4 users

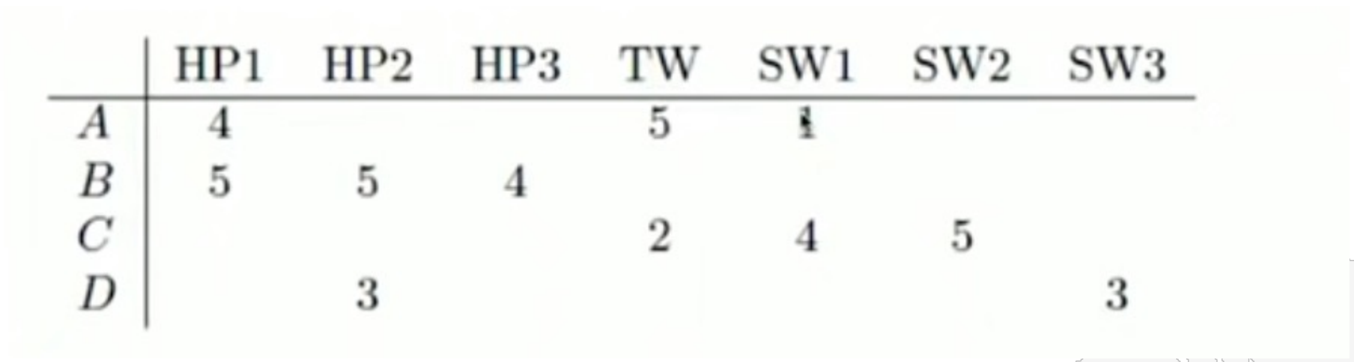


	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

HP:Harry Potter
SW:Star Wars
TW:Twilight

- **Consider users *x* and *y* with rating vectors *rx* and *ry***
  - *The rating vector of user B, r_b is (5,5,4,0,0,0,0)*
- **We need a similarity metric *sim(x,y) between rating vectors***
- **The metric should capture the intuition that *sim(A,B) > sim(A,C)***
  - *A and B both liked HP1, but A and C had very different opinions about TW and SW1*
- **Recall *sim(A,B)* = |*r_a intersect r_b*| / |*r_a union r_b* |** ***try Jaccard Similarity***
- ***sim (A,B) = 1/5; sim(A,C) = 2/4. since A&B rated only one movie in common***
  - *Conclusion: using Jaccard Similarity we get a result we don't want, namely*
  - *Sim(A,B) < Sim (A,C)*
- **Problem: ignores rating values**

Cosine Similarity

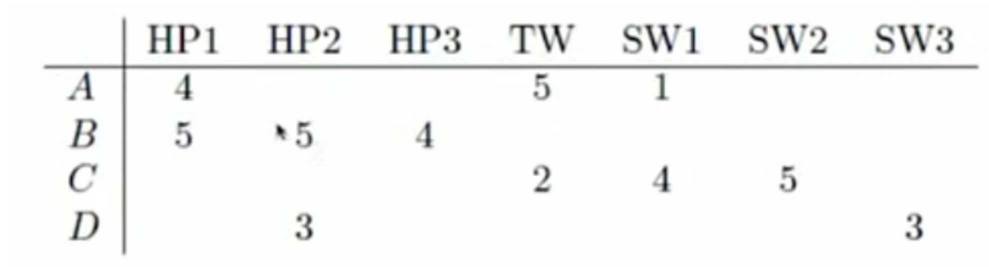


	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

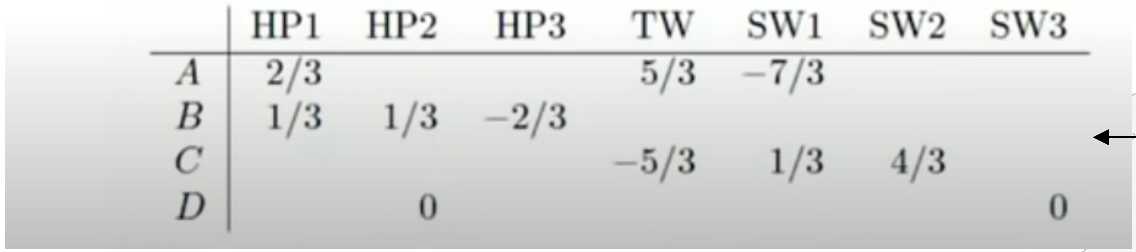
- *Instead of using Jaccard similarity, which ignored the rating values, let's use cosine similarity, the angle between the vectors; now we treat the unknown values as zero*
  - *sim (A, B) = cos(r_a, r_b)*
- *sim(A,B) = 0.38, sim(A,C) =0.32*
  - *Now sim(A,B) > sim(A,C), which is what we wanted,* **but not by much**
- **Problem: by treating missing ratings as zero, we sort of got the result we wanted, but actually the similarity of *A* and *C* should be farther apart than what we computed; using zero was not a great idea**

Centered Cosine Captures User Preferences

Solution: Normalize ratings by subtracting row mean



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

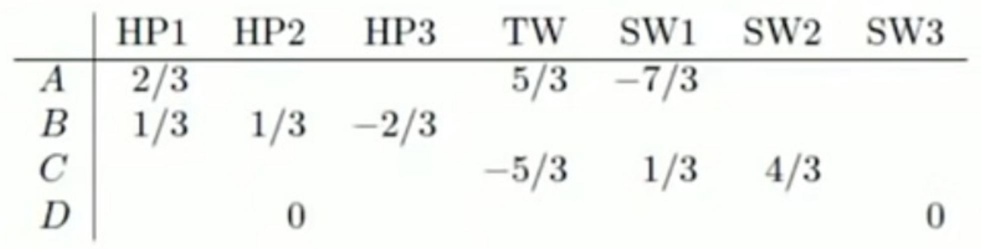


	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	2/3			5/3	−7/3		
B	1/3	1/3	−2/3				
C				−5/3	1/3	4/3	
D		0					0

- *The average rating for A is 10/3;*
- *The average rating for B is 14/3;*
- *The solution is to subtract the average rating for each user's score; e.g. user A and HP1 we get 4 – (10/3)= 2/3*
- *The resulting matrix*

$\longleftarrow$

- *sim(A,B) = cos(r_a,r_b) = 0.09; sim(A,C)=-0.56*
Result: A and C are very DISsimilar
- *sim(A,B) > sim(A,C)*



	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

- Captures intution better
  - Missing ratings treated as average
  - Handles "tough raters" and "easy raters"

Note: Summing the rows for any user gives zero, so positive ratings means they liked the movie
Another name for centered cosine is **Pearson Correlation**

Making Rating Predictions for a User

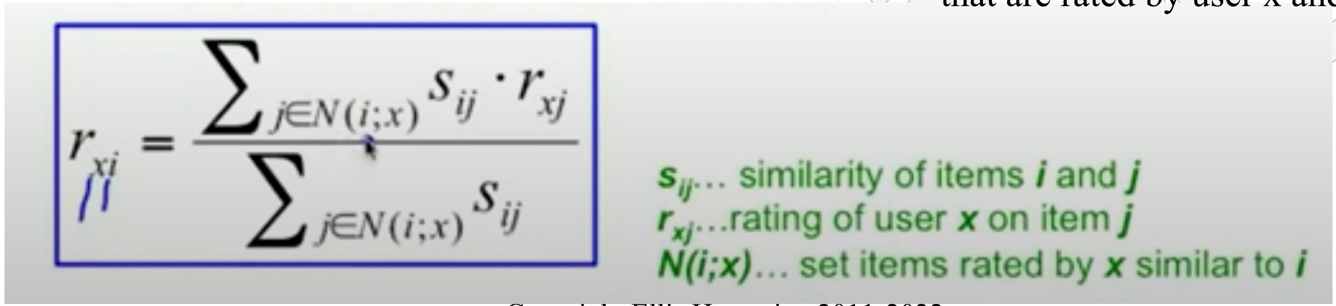
- Let r_x be the vector of user x's ratings
- Let N be the set of K users most similar to x who have also rated item i
- Prediction for user x and item i
- Option 1: $r_{xi} = 1/k \sum_{y \in N} (r_{yi})$ *use the average rating of users who rated i*
- Option 2: $r_{xi} = \sum_{y \in N} (s_{xy}\, r_{xy}) \,/\, \sum_{y \in N} s_{xy}$ **This is a weighted average**

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

Item-Item Collaborative Filtering

- So far: we have used cosine similarity for user-user collaborative filtering
- We try to apply it to content filtering
  - For item *i*, find other similar items
  - Estimate rating for item *i* based on ratings for similar items
  - Use the same similarity metrics and prediction functions as in user-user model

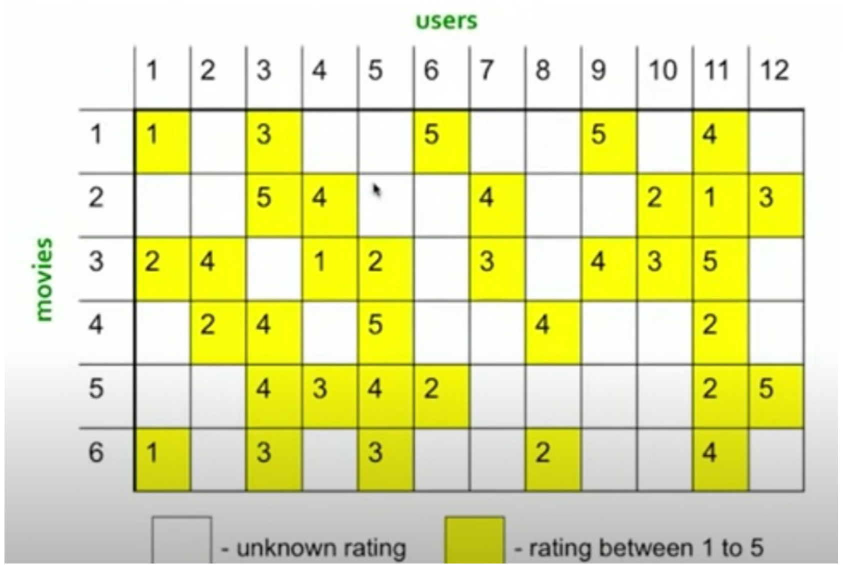
N(i;x) is the neighborhood of items that are rated by user x and similar to item i



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

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

Let's Do An Example
Item – Item Collaborative Filtering (|N| =2)

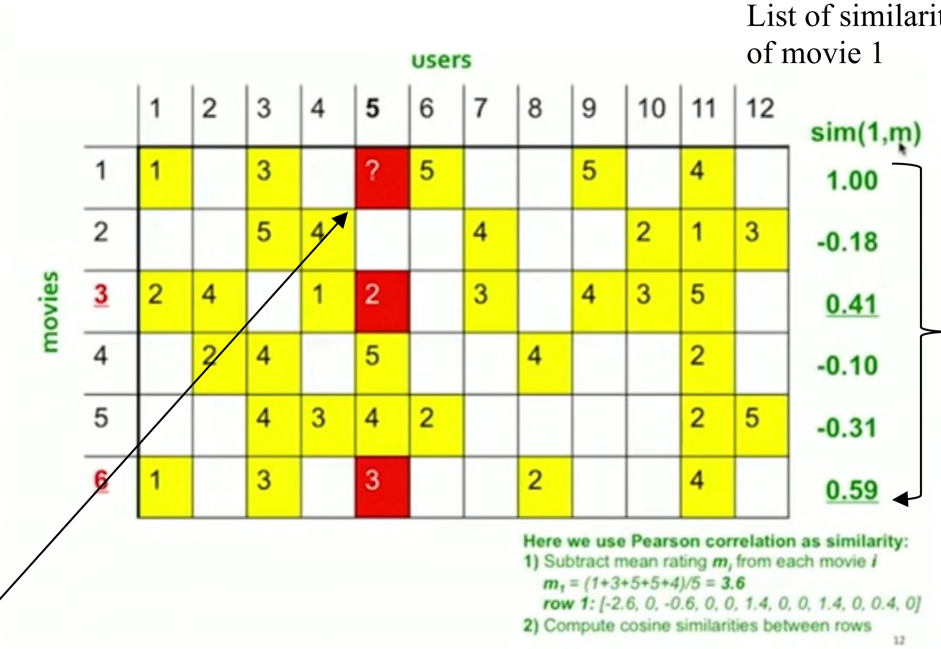


users

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

List of similarities of movie 1



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

Here we use Pearson correlation as similarity:
1) Subtract mean rating m_i from each movie i
   $m_1 = (1+3+5+5+4)/5 = 3.6$
   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

Goal: Estimate rating of movie 1 by user 5

N is 2, looking at the two nearest neighbors

Movie 3 and 6 are the two recommendations that are most similar to movie 1

Compute similarity weights

$S_{13} = 0.41$ $S_{16} = 0.59$

Use the weighted average to compute

$R_{15} = (0.41*2+0.59*3)/(0.41+0.59)=2.6$

Conclusion: user 5 will like movie 1: 2.6

Item-Item vs. User-User

- In theory user-user and item-item are dual approaches
- In practice, item-item outperforms user-user in many use cases
- Items are "simpler" than users
  - Items belong to a small set of "genres", users have varied tastes
  - *Item* similarity is more meaningful than *user* similarity

Collaborative filtering: Algorithm Complexity

- Expensive step is finding *k* most similar users (or items):*O(|U|)*
  - $|U|$ = size of utility matrix
- Too expensive to do at runtime
  - Could pre-compute
    - The set of similar items for every item
    - The set of similar users for every user
  - Naïve pre-computation takes time *O(n*|U|)* This is potentially a large number
    - Where *n* = number of users (items)
  - But we can use a previous technique, e.g.
    - Clustering

USC Viterbi School of Engineering

Problems with Collaborative Filtering

- Cold Start: There needs to be enough other users already in the system to find a match.
- Sparsity: If there are many items to be recommended, even if there are many users, the user/ratings matrix is sparse, and it is 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 tastes.
  - Tends to recommend popular items.

Combining Content and Collaboration

- Content-based and collaborative methods have complementary strengths and weaknesses.
- Combine methods to obtain the best of both.
- Various hybrid approaches:
  - Apply both methods and combine recommendations.
  - Use collaborative data as content.
  - Use content-based predictor as another collaborator.
  - Use content-based predictor to complete collaborative data.

Conclusions

- Recommending and personalization are important approaches to combating information overload.
- Machine Learning is an important part of systems for these tasks.
- Collaborative filtering has problems.
- Content-based methods address these problems (but have problems of their own).
- Integrating both is best.

USC Viterbi School of Engineering

Evaluation Metrics for Recommendation Engines

- *Recall* – the proportion of items that a user like that were recommended
  - t_p = number of recommended items
  - f_n = the remaining items

$$\text{Recall} = \frac{tp}{tp + fn}$$

- *Precision* – out of all recommended items, how many did the user like
  - t_p number of recommended items
  - $t_p + f_p$ the total items recommended

$$\text{Precision} = \frac{tp}{tp + fp}$$

- *Root Mean Squared Error* (RMSE)
  - Measures error in the predicted rating

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

- *Mean Reciprocal Rank*
  - The larger the MRR, the better the recommendation

$$\text{MRR} = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{1}{r(Q_i)}$$