GENERAL ASSEMBLY

RECOMMENDATION ENGINES

RECOMMENDATION SYSTEMS

What?

- Match users to products / items / brand / etc they have not experienced yet.
- Predict preferences based on past observations.

How?

- Produced by analysing similar user / item ratings to provide personalised recommendations to users.

Why?

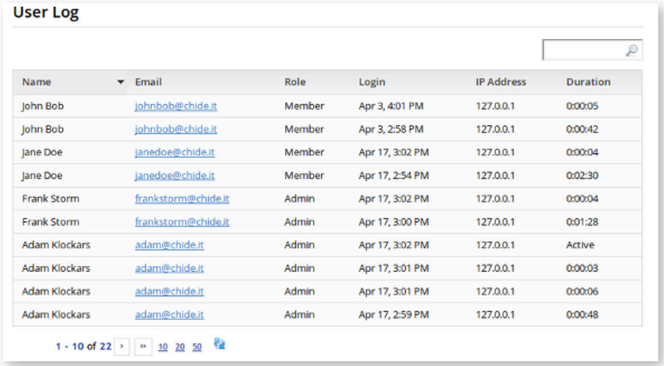
- Personalise UX → more $$$

DATA

WE NEED DATA TO RECOMMEND.

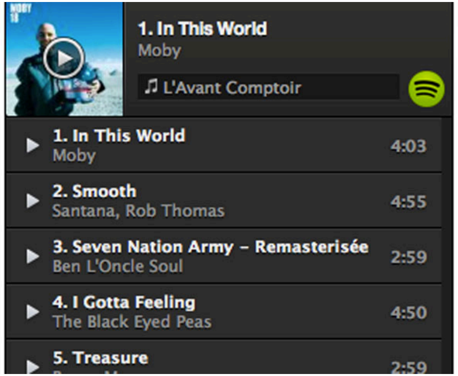
- Preferences
- Ratings
- Item meta-data
- User Behavior

EXAMPLES – TYPES OF DATA


User Log

Name	Email	Role	Login	IP Address	Duration
John Bob	johnbob@chide.it	Member	Apr 3, 4:01 PM	127.0.0.1	0:00:05
John Bob	johnbob@chide.it	Member	Apr 3, 2:58 PM	127.0.0.1	0:00:42
Jane Doe	janedoe@chide.it	Member	Apr 17, 3:02 PM	127.0.0.1	0:00:04
Jane Doe	janedoe@chide.it	Member	Apr 17, 2:54 PM	127.0.0.1	0:02:30
Frank Storm	frankstorm@chide.it	Admin	Apr 17, 3:02 PM	127.0.0.1	0:00:04
Frank Storm	frankstorm@chide.it	Admin	Apr 17, 3:00 PM	127.0.0.1	0:01:28
Adam Klockars	adam@chide.it	Admin	Apr 17, 3:02 PM	127.0.0.1	Active
Adam Klockars	adam@chide.it	Admin	Apr 17, 3:01 PM	127.0.0.1	0:00:03
Adam Klockars	adam@chide.it	Admin	Apr 17, 3:01 PM	127.0.0.1	0:00:06
Adam Klockars	adam@chide.it	Admin	Apr 17, 2:59 PM	127.0.0.1	0:00:48

1 - 10 of 22 10 20 50



Ratings
Upvotes / Downvotes
Weighted Scale
Grades
Relevance Feedback

Access Logs
Session Lengths
Time spent on a page
Clicks / Non-Clicks
Purchase History
Product Descriptions

Listening History
Playlist Creates
Follows / Unfriend
Impressions
Email Reads / Impressions

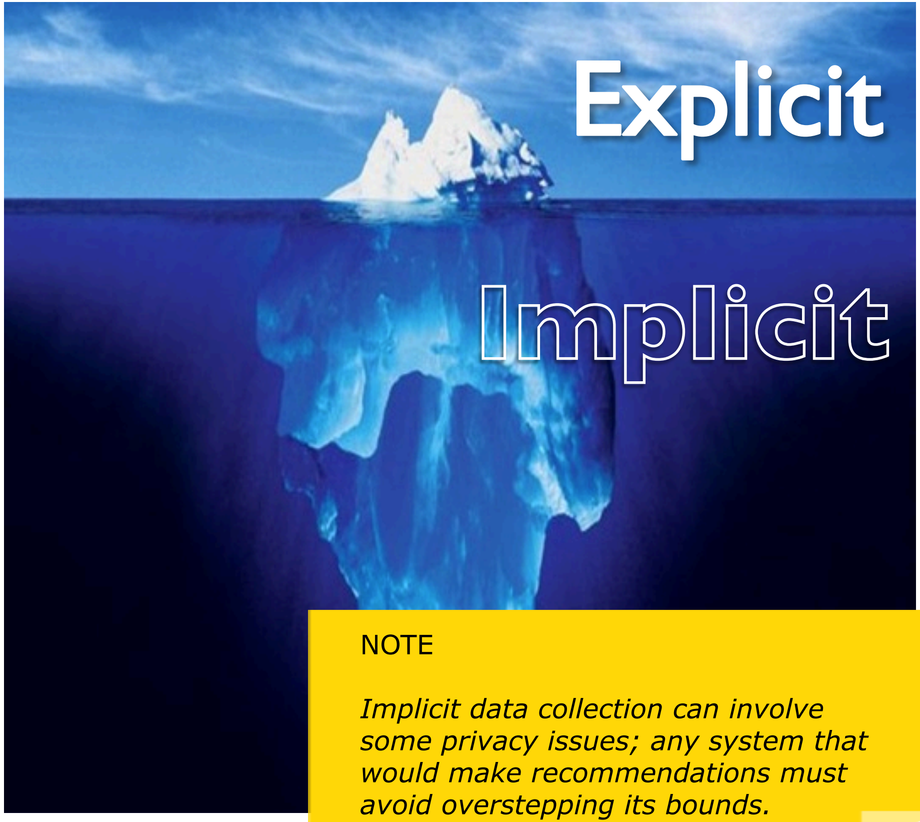
EXAMPLES – TYPES OF DATA

Explicit

- i.e. ratings, surveys, reviews
- Easy to interpret
- Expensive

Implicit

- Activity logs, clicks, impressions
- Hard to interpret
- Cheap



NOTE

Implicit data collection can involve some privacy issues; any system that would make recommendations must avoid overstepping its bounds.

Explicit Feedback

- Frequently in the form of ratings
- Granularly represents preferences
- Requires extra effort from the user

Explicit Feedback – Considerations

- Consistent scale for all ratings
- Can ratings be skewed by self/selection-bias
- When the data was collected (before or after experience)
- Context of presentation

Implicit Feedback

- Make recommendations when no rating data is explicitly collected from a user.
- Convert user behavior into user preferences
- Challenge: How exactly does one infer preference based on actions in a system?

IMPLICIT DATA / FEEDBACK - GENERAL

Implicit feedback is everywhere.

- Email impressions
- Email click-throughs
- Conversions
- Demographic
- Session lengths
- Login attempts
- Track plays
- Money spent

- Ad impressions
- Ad clicks
- Ad click-purchase
- Web “click depth”
- # of swipes
- Profile views
- Message initiations
- Poll Votes

- Friend / unfriend
- Follow / unfollow
- *Like
- Post text
- Image EXIF
- Friends in common
- Message text
- Food purchases

- Geospatial data
- Store cameras
- Wifi logins / MAC
- Time series
- Objects in photos
- Driving record
- Credit history
- Topics most read

Implicit Feedback Caveats: Question Everything

- Preferences can be vague
- May need to process tons of data to get what you want
- Analysis can be complicated / meaning hard to find
- Users don’t tell you what you want to know
- Easy to project bias onto data
- Positive / negative experience hard to assess

Implicit + Explicit Feedback work together

If a user rates an item, use implicit feedback to **validate credibility**

- Did they read the article?
- Do they own the item?
- Did they rate before or after experience?
- Do other users mention them?
- Does user tend to rate high or low?
- How likely was the rating automated?

Use implicit data to **understand the context and characteristics of a rating**.

- Does time of day affect rating?
- Which kinds of reviews do they typically write?
- Are the reviews positive or negative?
- Do other users like their reviews?

Explicit

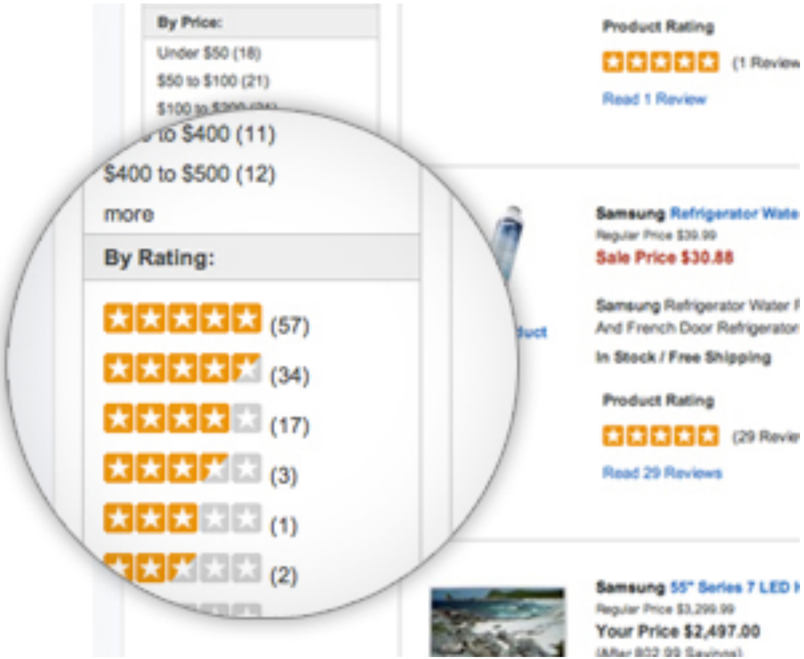
- Higher value with respect to preferences
- Usually collected as a “rating”
- Collection is responsibility of user
- More direct evaluation of items

Implicit

- Easy to collect in large quantities
- More difficult to work with
- Assumes nothing about the user (could be anyone!)
- Goal is to convert into preferences

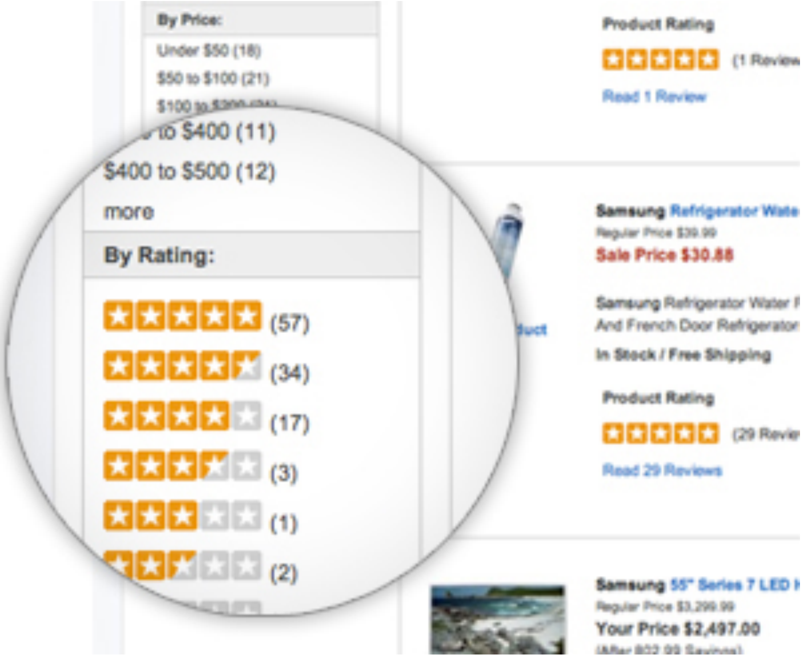
Explicit or Implicit?

EXAMPLES – TYPES OF DATA



Explicit or Implicit?

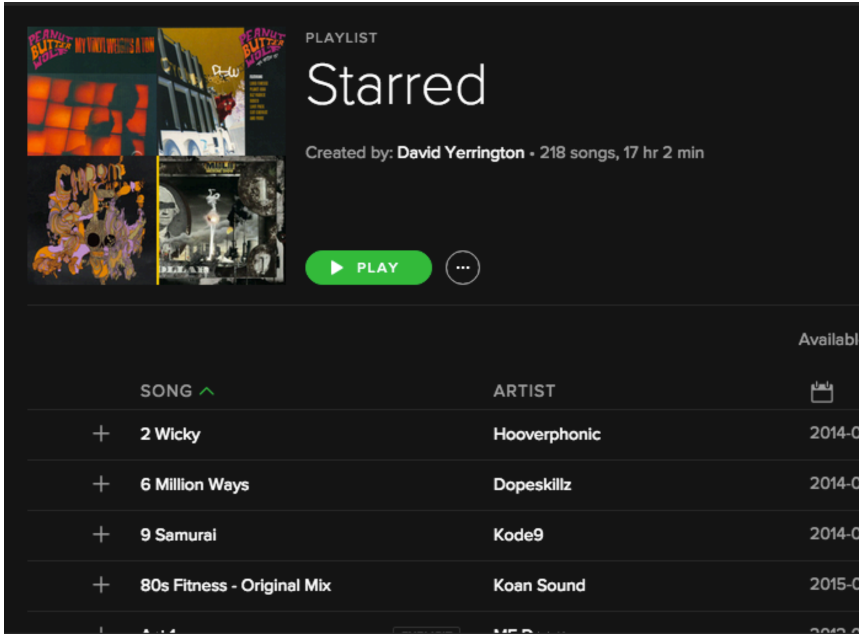
EXAMPLES – TYPES OF DATA



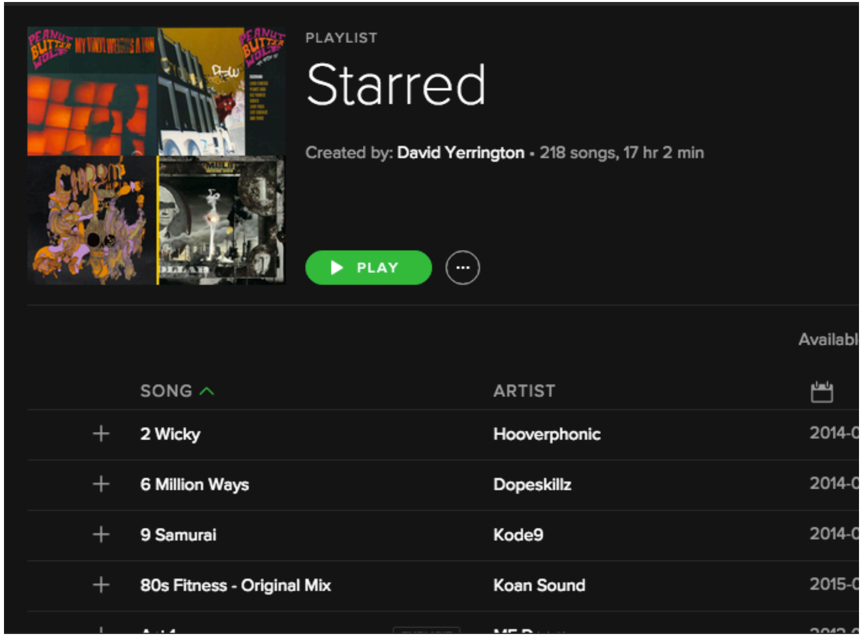
Explicit or Implicit?

Ratings: *Explicit*

EXAMPLES – TYPES OF DATA



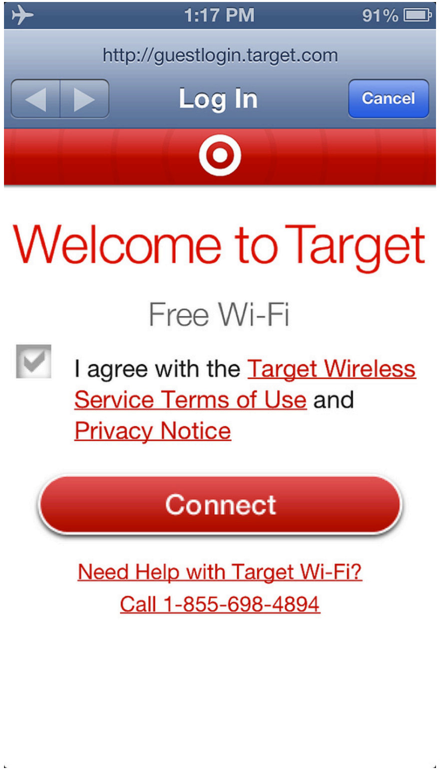
Explicit or Implicit?



Explicit or Implicit?

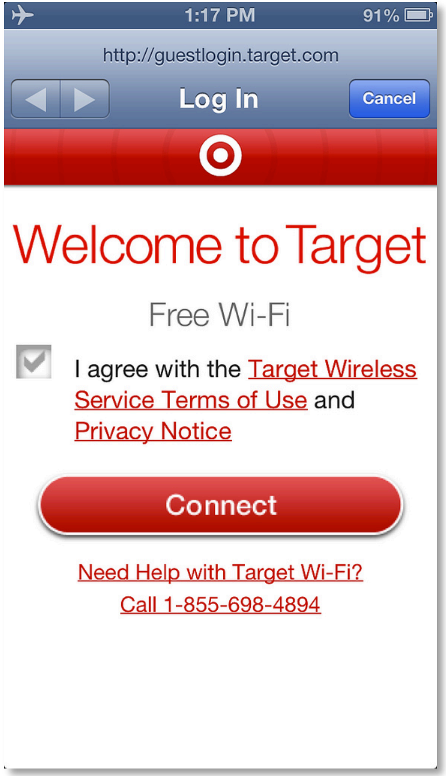
Both!

EXAMPLES – TYPES OF DATA



Explicit or Implicit?

EXAMPLES – TYPES OF DATA



Explicit or Implicit?

Wifi logs: *Implicit*!

GENERAL

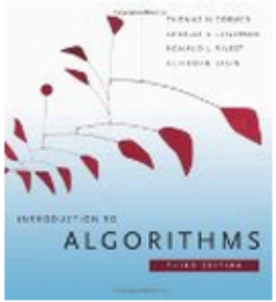
In **content-based filtering**, items are mapped into a feature space, and recommendations depend on item characteristics.

Collaborative filtering assumes users who have similar preferences in the past are likely to have similar preferences in the future.

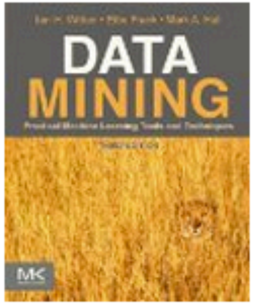
EXAMPLES – AMAZON CONTENT-BASED

Recommendations for You in Books

Cracking the Coding Interview: 150...
› Gayle Laakmann McDowell
Paperback
★★★★★ (166)
~~$39.95~~ $23.22
Why recommended?

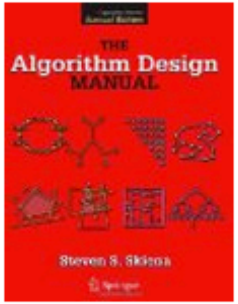


Introduction to Algorithms
Thomas H. Cormen, Charles E...
Hardcover
★★★★☆ (85)
~~$92.00~~ $80.00
Why recommended?



Data Mining: Practical Machine...
› Ian H. Witten, Eibe Frank, Mark A. Hall
Paperback
★★★★☆ (27)
~~$69.95~~ $42.09
Why recommended?

Elements of Programming Interviews...
› Amit Prakash, Adnan Aziz, Tsung-Hsien Lee
Paperback
★★★★½ (25)
~~$29.99~~ $26.18
Why recommended?



The Algorithm Design Manual
› Steve Skiena
Paperback
★★★★½ (47)
~~$89.95~~ $71.84
Why recommended?

EXAMPLES – YOUTUBE

Recommended for you because you watched
Sugar Minott - Oh Mr Dc (Studio One)

Mikey Dread - Roots and Culture
by klaxonklaxon · 1,164,133 views
Lyrics:
Now here comes a special request
To each and everyone

Recommended for you because you watched
Thelonious Monk Quartet - Monk In Denmark

Bill Evans Portrait in Jazz (Full Album)
by hansgy1 · 854,086 views
Bill Evans Portrait in Jazz 1960
1. Come Rain or Come Shine - 3.19 (0:00)
2. Autumn Leaves - 5.23 (3:24)

Recommended for you because you watched
Bob Marley One Drop

Bob Marley - She's gone
by Dionysios29 · 1,058,704 views
This is one of the eleven songs of album Kaya that Bob Marley and The Wailers creative in 1978.
Lyrics:

EXAMPLES – NYTIMES.COM

MOST E-MAILED | RECOMMENDED FOR YOU

1. **How Big Data Is Playing Recruiter for Specialized Workers**
2. SLIPSTREAM
   When Your Data Wanders to Places You've Never Been
3. MOTHERLODE
   The Play Date Gun Debate
4. **For Indonesian Atheists, a Community of Support Amid Constant Fear**
5. **Justice Breyer Has Shoulder Surgery**
6. BILL KELLER
   Erasing History

Collaborative

or

Content based?

8. How do you determine my Most Read Topics?

Back to top

Each NYTimes.com article is assigned topic tags that reflect the content of the article. As you read articles, we use these tags to determine your most-read topics.

To search for additional articles on one of your most-read topics, click that topic on your personalized Recommendations page. To learn more about topic tags, visit Times Topics.

CONTENT-BASED FILTERING

- Map each item into a **feature space**: Users and items are represented by vectors in this space
- **Item vectors** measure the degree to which the item is described by each feature.
- **user vectors** measure a user’s preferences for each feature.
- Ratings are generated by taking **dot products** of user & item vectors.

EXAMPLE – CONTENT-BASED FILTERING

features = (big box office, aimed at kids, famous actors)

Items (movies):	**Prediction (for Alice)**	
Finding Nemo = (5, 5, 2)	5*-3 + 5*2 + 2*-2	= -9
Mission Impossible = (3, -5, 5)	3*-3 + -5*2 + 5*-2	= -29
Jiro Dreams of Sushi = (-4, -5, -5)	-4*-3 + -5*2 + -5*-2	= **+12**

User:

Alice = (-3, 2, -2)

EXAMPLE – CONTENT-BASED FILTERING

features = (big box office, aimed at kids, famous actors)

Items (movies):	Prediction (for Bob)	
Finding Nemo = (5, 5, 2)	5*4 + 5*-3 + 2*5	= +15
Mission Impossible = (3, -5, 5)	3*4 + -5*-3 + 5*5	= **+52**
Jiro Dreams of Sushi = (-4, -5, -5)	-4*4 + -5*-3 + -5*5	= -26

User:

Bob = (4, -3, 5)

Pandora

- Maps songs into a feature space using features (or “genes”)
- Using song vectors that depend on these features, Pandora creates a station with similar music.

TF-IDF

- Create document profiles as weighted vectors of its tags
- Combine those with ratings to create user profiles

http://www.music-map.com/

CONTENT-BASED FILTERING

- Must map items into a feature space
- Recommendations are limited in scope → no serendipitous discoveries
- Hard to create cross-content recommendations (e.g. books/music films) → would require comparing elements from different feature spaces!
- Needs well structured data

ASSOCIATION RULES - METRICS

{X} → {Y} (People who liked X also liked Y)

Metrics:

Support: Default popularity of an item

$$\text{Support (X)} = \frac{\text{Transactions containing (X)}}{\text{Total Transactions}}$$

Confidence: Likelihood that item Y is bought if X is bought

$$\text{Confidence (X}\rightarrow\text{Y)} = \frac{\text{Number of Transactions (X\&Y)}}{\text{Number of Transactions (X)}}$$

Lift: Increase in ratio of sales of Y when X is sold

$$\text{Lift (X}\rightarrow\text{Y)} = \frac{\text{Support (X \& Y)}}{\text{Support (X) * Support (Y)}}$$

APRIORI ALGORITHM IN PYTHON

	user_id	movie_id	movie title
0	196	242	Kolya (1996)
1	186	302	L.A. Confidential (1997)
2	22	377	Heavyweights (1994)
3	244	51	Legends of the Fall (1994)
4	166	346	Jackie Brown (1997)

	user_id	movie_views
0	1	[Three Colors: White (1994), Grand Day Out, A ...
1	2	[Rosewood (1997), Shall We Dance? (1996), Star...
2	3	[How to Be a Player (1997), Devil's Own, The (...
3	4	[Mimic (1997), Ulee's Gold (1997), Incognito (...
4	5	[GoldenEye (1995), From Dusk Till Dawn (1996),...

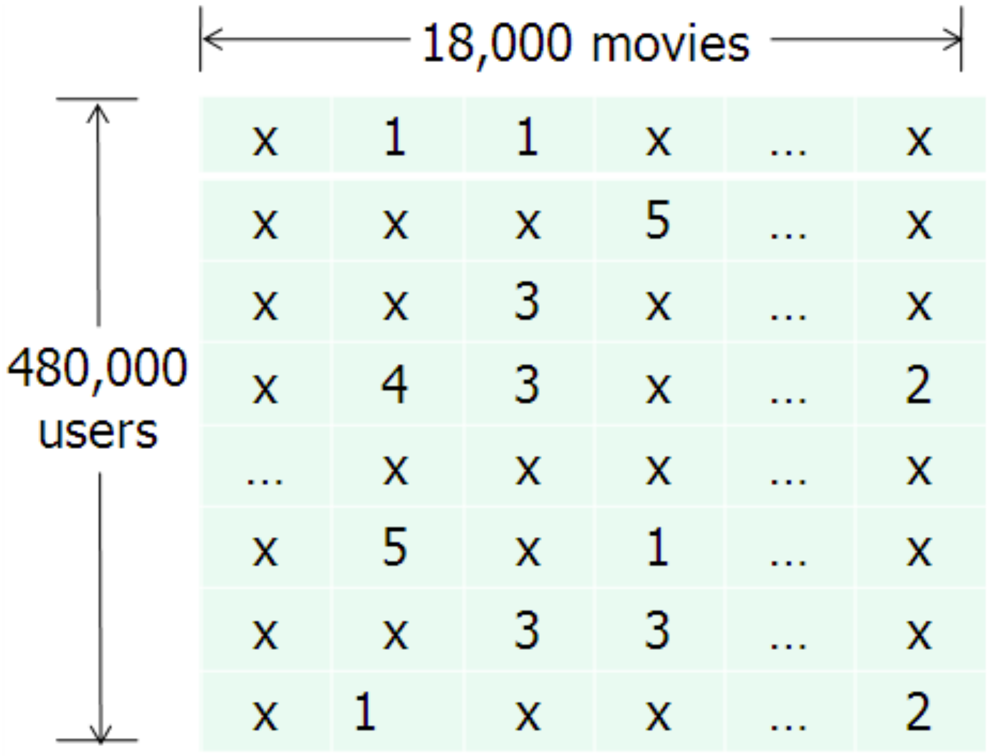
```python
from apyori import apriori

df = df.groupby(['user_id'])['movie title'].apply(
    lambda x: x.values.tolist()).reset_index(name='movie_views')

df_listoflists=[]
for row in df.movie_views:
    df_listoflists.append(list(row))

association_rules = apriori(df_listoflists,
                            min_support=0.2,
                            min_confidence=0.1,
                            min_lift=3,
                            max_length=2)
association_results = list(association_rules)

for item in association_results:

    pair = item[0]
    print(pair)
    items_list = [x for x in pair]
    print("Rule: " + items_list[0] + " -> " + items_list[1])
    print("Support: " + str(item[1]))
    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("=====================================")
```

```
frozenset({'20,000 Leagues Under the Sea (1954)', '12 Angry Men (1957)'})
Rule: 20,000 Leagues Under the Sea (1954) -> 12 Angry Men (1957)
Support: 0.2
Confidence: 1.0
Lift: 5.0
=====================================
```

COLLABORATIVE FILTERING

COLLABORATIVE FILTERING

- For given user find k most similar users
- Only interested in the existing user-item ratings themselves
- Dataset is ratings matrix with columns corresponding to items, and rows corresponding to users
- Creates recommendations based on other users with similar tastes
- Cold start problem: Until users rate several items, we don't know anything about their preferences!



Sparse Matrix

source: http://www.eecs.berkeley.edu/~zhanghao/main/publications/subfolder/netflix.png

ITEM-BASED COLLABORATIVE FILTERING

Jaccard Similarity:

→ Typically used where products don’t have numeric ratings

Cosine Similarity:

→ Use for sparse data

Pearson Similarity:

→ Use when data is subject to user-bias/different rating scales

Jaccard Similarity:

Defines similarity between two sets of objects

Number of similar elements

$$JS(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|}$$

Number of distinct elements

$$JS(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|}$$

$$JS(\{1, 2, 3\}, \{2,3,4\}) = \frac{\{2, 3\}}{\{1, 2, 3, 4\}} = \frac{2}{4}$$

$$JS(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|}$$

User one: {“Target”, “Banana Republic”, “Old Navy”}
User two: {“Banana Republic”, “Gap”, “Kohl’s”}

JS (User one, User two) =

NETFLIX PRIZE

THE NETFLIX PRIZE

- Competition (2006-2009) to make a 10% RMSE improvement to Netflix’s recommendation system.
- Grand prize was $1m dollars.
- Ratings matrix had >100mm numerical entries (1-5 stars) from ~500k users across ~17k movies.
- Winning entry was a stacked ensemble of 100’s of models (including neighborhood & matrix factorization models) that were blended using boosted decision trees.
- Winning strategy came down to last-minute team mergers & creative blending schemes to shave 3rd & 4th decimals off RMSE (concerns that would not be important in practice).