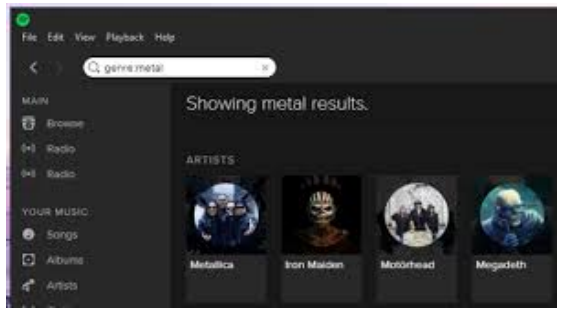
Movie Recommendation – Collaborative Filtering

CMPT 733

Spring 2017

Apala Guha



Users

Preferences
- rating, vote, follow
- review
- click, time spent

Recommendations
- ranking
- related to context/mood
- sequence

Products
- books, movies
- friends, people to follow
- playlists, service bundles

	Alice	Bob	Carol	Dave
Shanghai Triad	5	?	4	?
Usual suspects	?	?	1	1
In Love and War	2	3	2	1
Anna Karenina	2	5	?	?
Incognito	1	?	2	3

Recommender

- demographics

- index no.

- aggregate stats

Non-personalized recommender

Non-personalized

- Nuances not known or required
- Initial recommendation/profile-building stage
- The same recommendation fits many

- demographics
- Per-user model

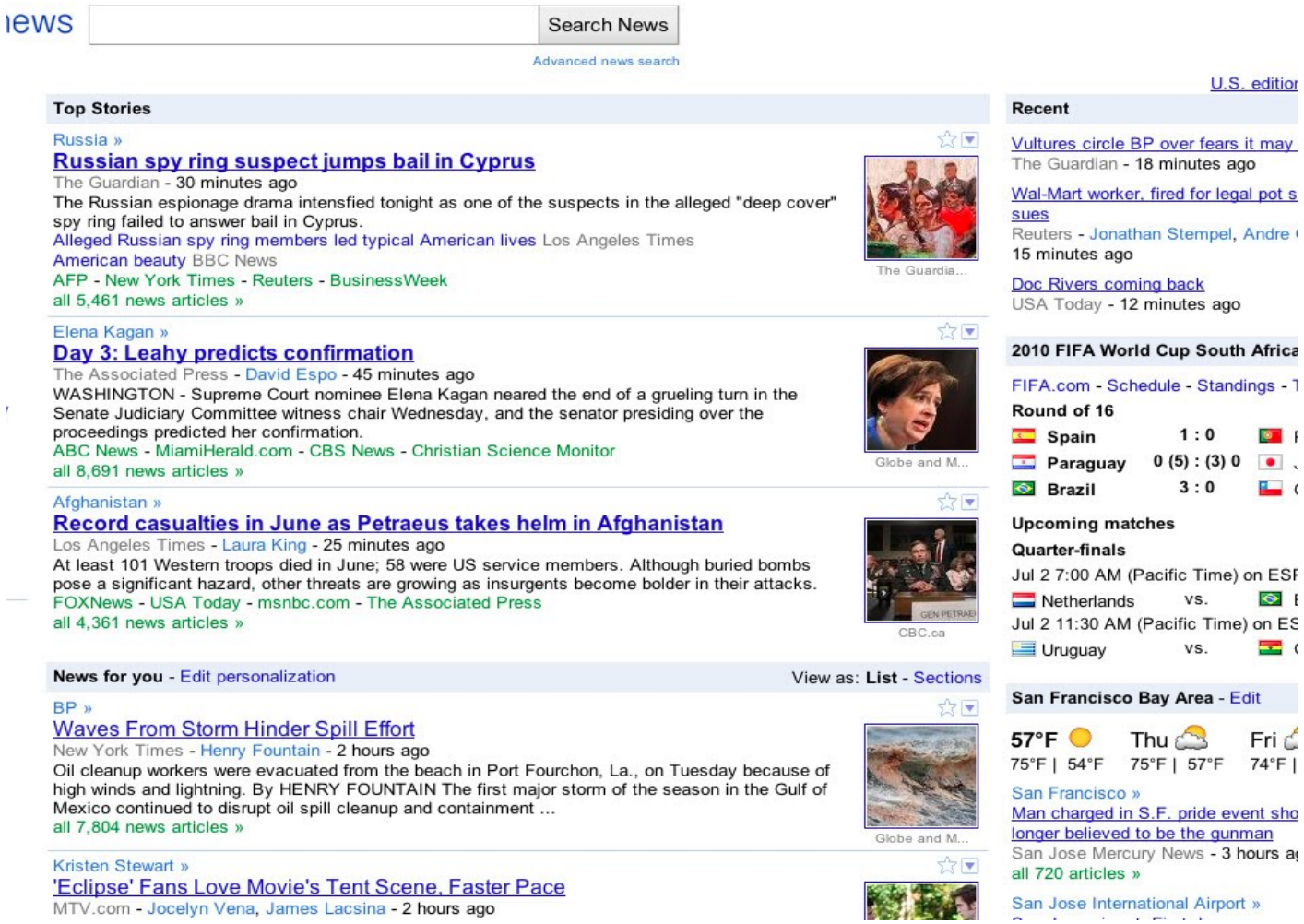
- index no.
- Stable feature vector

- aggregate stats
- user-item stats

Content-based filtering

CBF

- Stable item description e.g. publisher summary
- Item feature vectors e.g. TF-IDF
- Learn per-user models to map item feature vectors => user ratings
- Assumes user preferences remain stable and are not highly nuanced
- Product attributes suitably captured by summary/review



- demographics
- Per-user model
- User attributes unknown

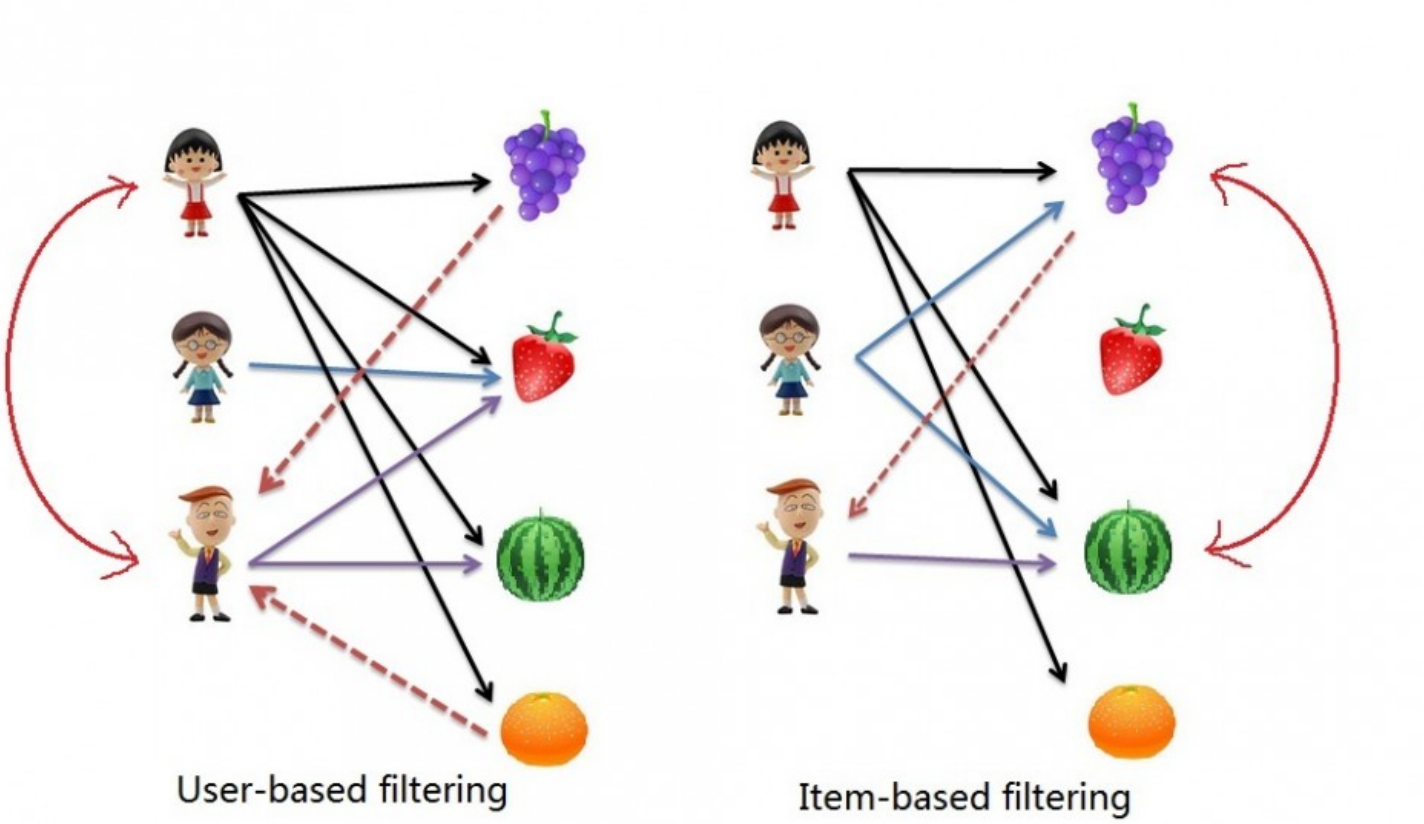
- index no.
- Stable feature vector
- Item attributes unknown

- aggregate stats
- user-item stats

Nearest-neighbor Collaborative filtering

Nearest-neighbor CF

- User-user CF: find the users most similar to this user and how these similar users rated the item
- Item-item CF: find the items most similar to this item and how this user rated these similar items
- No assumed attributes – appropriate when it is difficult to manually enumerate attributes of items



User-user CF

- We want to find $r_{u,i}$
- similarity between two users = $w_{u1,u2}$ = $similarity_i(R_{u1,i}, R_{u2,i})$
  - R is the vector of items that both users rated
  - Pearson/rank/Jaccard/cosine
- Discard users who are not similar – their opinion is not important
- $r_{u,i} = sum_{u'}(r_{u',i} * w_{u,u'}) / sum_{u'}(w_{u,u'})$
- Correct for user rating scale: compute deviation from user mean
  $r_{u,i} = r_u + sum_{u'}(d_{u',i} * w_{u,u'}) / sum_{u'}(w_{u,u'})$

Item-item CF

- We want to find $r_{u,i}$
- similarity between two items = $w_{i1,i2}$ = $similarity_u(R_{u,i1}, R_{u,i2})$
  - R is the vector of ratings by users that rated both items
  - Pearson/rank/Jaccard/cosine
- Discard items that are not similar
- $r_{u,i} = sum_{i'}(r_{u,i'} * w_{i,i'}) / sum_{i'}(w_{i,i'})$
- Correct for user rating scale: compute deviation from user mean
  $r_{u,i} = r_u + sum_{i'}(d_{u,i'} * w_{i,i'}) / sum_{i'}(w_{i,i'})$
- Which one is better and why?

Issues

- Typically user set has much higher cardinality than item set
- Items on average have more ratings than users
  - Therefore item similarity is more meaningful
  - Also there are fewer pairs to calculate

- demographics
- Per-user model
- User attributes unknown
- Latent user attributes

- index no.
- Stable feature vector
- Item attributes unknown
- Latent item attributes

- aggregate stats
- user-item stats

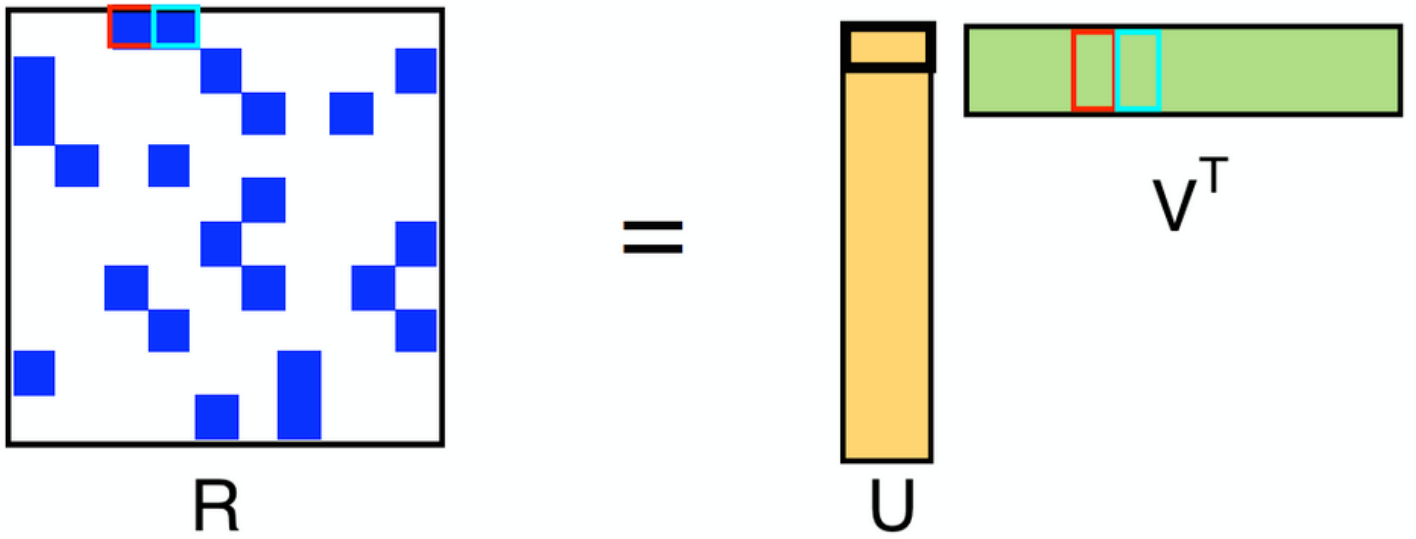
Matrix Factorization-based Collaborative filtering

	Alice	Bob	Carol	Dave
Shanghai Triad	5	?	4	?
Usual suspects	?	?	1	1
In Love and War	2	3	2	1
Anna Karenina	2	5	?	?
Incognito	1	?	2	3

$$M_{U*I} = X_{U*K} * Y^{T}_{I*K}$$

Matrix factorization-based CF

- K = #latent attributes
- Once factorized, any user-item pair can be predicted by multiplying the user vector with the item vector
- Similar to regression learning, known matrix cells are used as examples to learn the latent attributes
- However there are two sets of attributes to learn (X and Y) – learning starts with initial guesses for both X and Y, current values of X are used to drive each step in learning Y values and vice-versa => known as alternating least squares (*ALS*) method
- Why use it?



Matrix-factor CF

- Computationally efficient
- Users may not be entirely similar to each other, so also for items – some users may be similar on some aspects but differ on others, some products may appear similar to some users but different to others
- Dimensionality reduction for high-dimension problems e.g. in Instagram the user set and item set have the same cardinality

Assignment

- Use PySpark ML recommendation module for building ALS models
- Use PySpark SQL grouping and aggregation API for item-item CF