CS6220: DATA MINING TECHNIQUES

Recommender Systems

Instructor: Yizhou Sun

yzsun@ccs.neu.edu

March 22, 2016

Methods to Learn

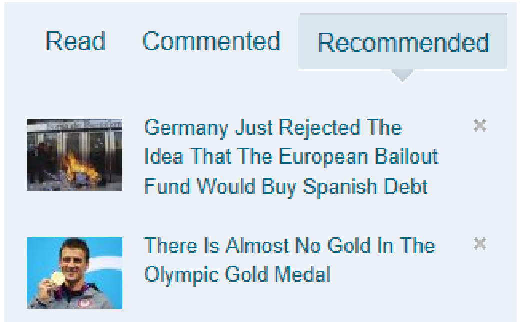
	Matrix Data	Text Data	Set Data	Sequence Data	Time Series	Graph & Network	Images
Classification	Decision Tree; Naïve Bayes; Logistic Regression SVM; kNN			HMM		Label Propagation*	Neural Network
Clustering	K-means; hierarchical clustering; DBSCAN; Mixture Models; kernel k-means*	PLSA				SCAN*; Spectral Clustering	
Frequent Pattern Mining			Apriori; FP-growth	GSP; PrefixSpan			
Prediction	Linear Regression				Autoregression	**Recommendation**	
Similarity Search					DTW	P-PageRank	
Ranking						PageRank	

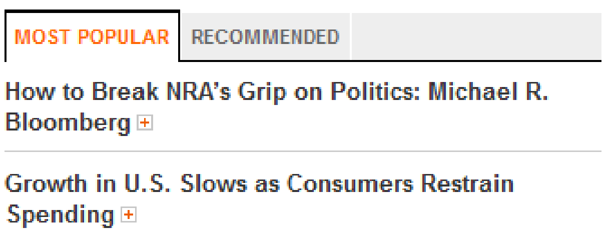
Recommender Systems

- What is Recommender System?
- Collaborative Filtering
- Content-based Recommendation
- Evaluation Metrics
- Summary

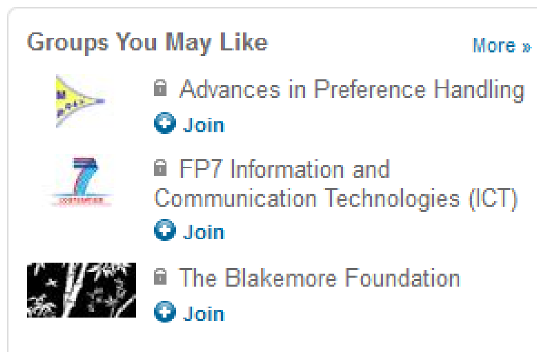
Recommender Systems

- Application areas





In the Social Web



Why using Recommender Systems?

- Value for the customer
  - Find things that are interesting
  - Narrow down the set of choices
  - Help me explore the space of options
  - Discover new things
  - Entertainment
  - …
- Value for the provider
  - Additional and probably unique personalized service for the customer
  - Increase trust and customer loyalty
  - Increase sales, click trough rates, conversion etc.
  - Opportunities for promotion, persuasion
  - Obtain more knowledge about customers
  - …

Representation

- Sparse Matrix
  - Explicit Feedback

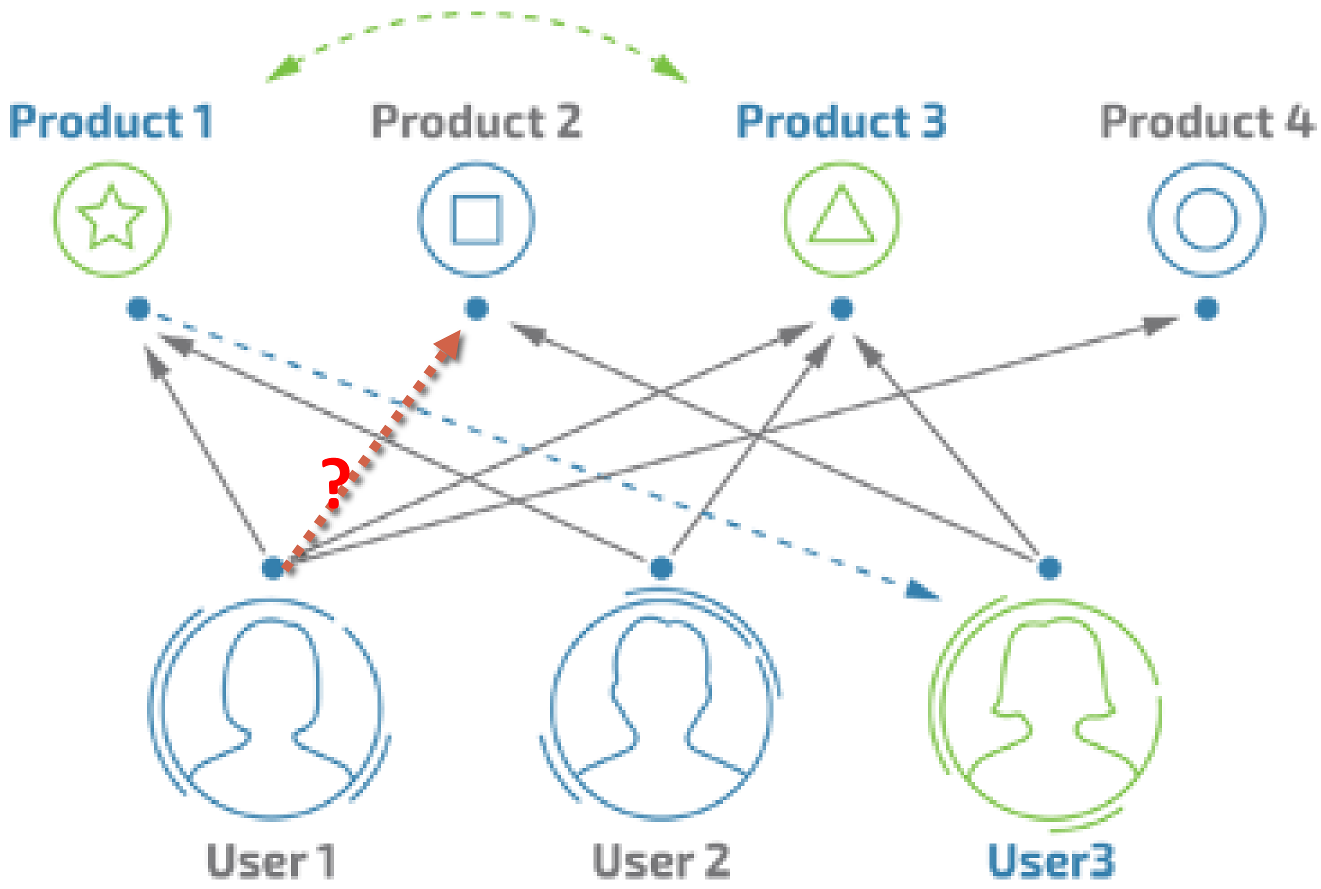
Users	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	⋯
User1	?	?	4	?	1	?	⋯
User2	2	5	2	?	?	2	⋯
User3	?	?	5	3	2	4	⋯
User4	1	?	?	4	?	?	⋯
User5	2	3	?	?	?	?	⋯
⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯

- Implicit Feedback

		Items				
						...
Users	Alice	1	1	0	0	
	Bob	0	0	1	1	
	Corey	1	0	1	0	
	...					

A Network Point of View

- Link prediction problem



Methods

- Collaborative filtering
- Content-based recommendation
- Hybrid methods

Recommender Systems

- What is Recommender System?
- Collaborative Filtering
- Content-based Recommendation
- Evaluation Metrics
- Summary

Collaborative Filtering (CF)

- The most prominent approach to generate recommendations
  - used by large, commercial e-commerce sites
  - well-understood, various algorithms and variations exist
  - applicable in many domains (book, movies, DVDs, ..)
- Approach
  - use the "wisdom of the crowd" to recommend items

- Basic assumption and idea
  - Users give ratings to catalog items (implicitly or explicitly)
  - Customers who had similar tastes in the past, will have similar tastes in the future

User-based Collaborative Filtering

1. Define similarity between users according to the history matrix
2. Decide how many "peers" to consider
3. Use peers' ratings to predict the rating between an active user and an item

	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

(1) Define Similarities between Users

- Pearson correlation between user a and b

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

  - $r_{a,p}$: *rating of user a to item p*
  - P: *a set of items that are rated by both a and b*
  - $\overline{r_a}, \overline{r_b}$: *average rating of user a and b*

- Or, $sim(a,b) = \frac{cov(r_a, r_b)}{\sigma(r_a)\sigma(r_b)}$
  - $cov(r_a, r_b)$: covariance between a and b
  - $\sigma(r_a), \sigma(r_b)$: standard deviation of a and b

Example

- $sim(Alice, User1)$
  - $\overline{r_{Alice}} = \frac{5+3+4+4}{4} = 4; \sigma(Alice) = 0.707$
  - $\overline{r_{User1}} = \frac{3+1+2+3}{4} = 2.25; \sigma(User1) = 0.9574$
  - $cov(Alice, User1) = 0.6667;$
  - $\boldsymbol{\Rightarrow sim(Alice, User1) = \frac{0.6667}{0.707 * 0.9574} = 0.8528}$

	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 = 0.85
sim = 0.70
sim = -0.79

(2) Decide how many peers to use

- Usually only use top K most similar users for prediction
  - i.e., based on top-K most similar users' rating for an item

(3) Predict the rating

- A common prediction function:

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

- Calculate, whether the neighbors' ratings for the unseen item *i* are higher or lower than their average
- Combine the rating differences – use the similarity as a weight
- Add/subtract the neighbors' bias from the active user's average and use this as a prediction

Example

- Use top-2 neighbor for prediction
  - Alice's top-2 neighbor are User1 and User2
  - $pred(Alice, Item5) = \overline{r_{Alice}} + \frac{sim(Alice, User1)(r_{User1,Item5} - \overline{r_{User1}}) + sim(Alice, User2)(r_{User2,Item5} - \overline{r_{User2}})}{sim(Alice, User1) + sim(Alice, User2)}$

$$= 4 + \frac{0.85*(3-2.25)+0.70*(5-3.5)}{0.85+0.70} = 5.0887$$

	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

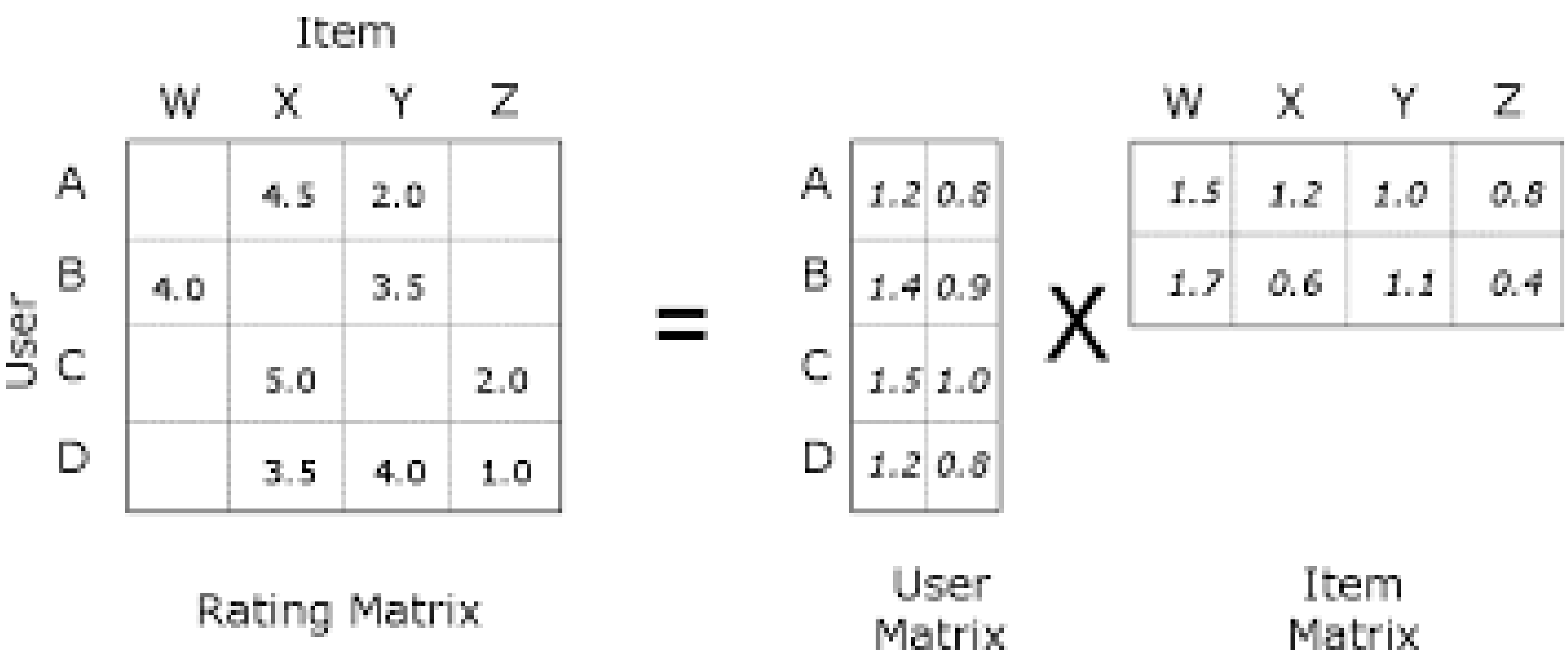
sim = 0.85
sim = 0.70

Model-based Collaborative Filtering

- User-based CF is said to be "memory-based"
  - the rating matrix is directly used to find neighbors / make predictions
  - does not scale for most real-world scenarios
  - large e-commerce sites have tens of millions of customers and millions of items
- Model-based approaches
  - based on an offline pre-processing or "model-learning" phase
  - at run-time, only the learned model is used to make predictions
  - models are updated / re-trained periodically
  - large variety of techniques used
  - model-building and updating can be computationally expensive

Matrix Factorization for Recommendation

- Map users and items into the same latent space



Now users and items are comparable

- Recommendation: find items that are close to users in the new space

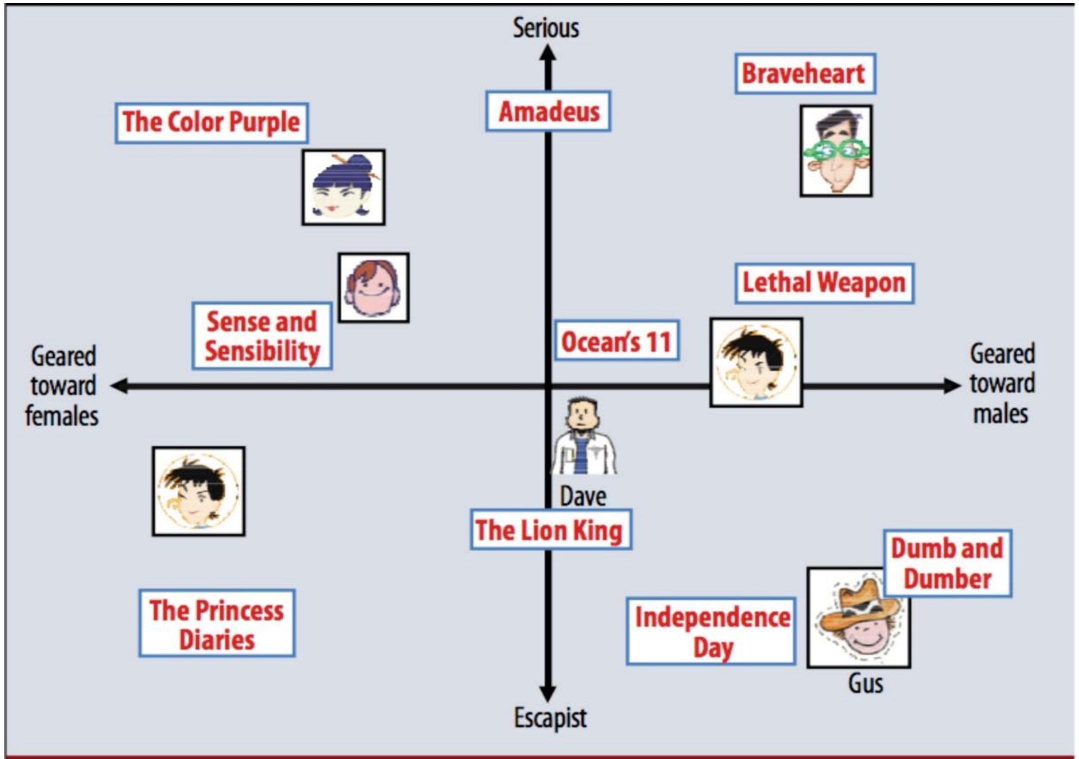


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

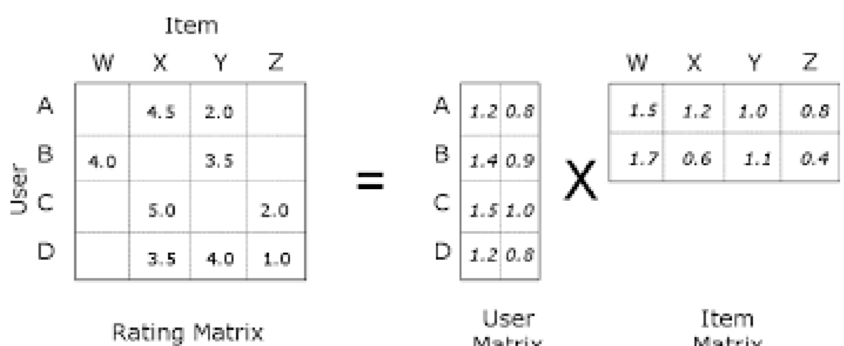
Procedure

- Training stage
  - Use existing matrix to learn the latent feature vector for both users and items by matrix factorization
- Recommendation stage
  - Predict the score for unknown (user, item) pairs

Training Stage

- r_{ui}: the rating from u to i
- p_u: the latent feature vector for user u
- q_i: the latent feature vector for item I
- $\hat{r}_{ui}$: score function for (u,i), $\hat{r}_{ui} = q_i^T p_u$
- Objective function:

$$\min_{p^*,q^*} \sum_{(u,i)\in D} \left(r_{ui} - q_i^T p_u\right)^2 + \lambda\left(||q_i||^2 + ||p_u||^2\right)$$



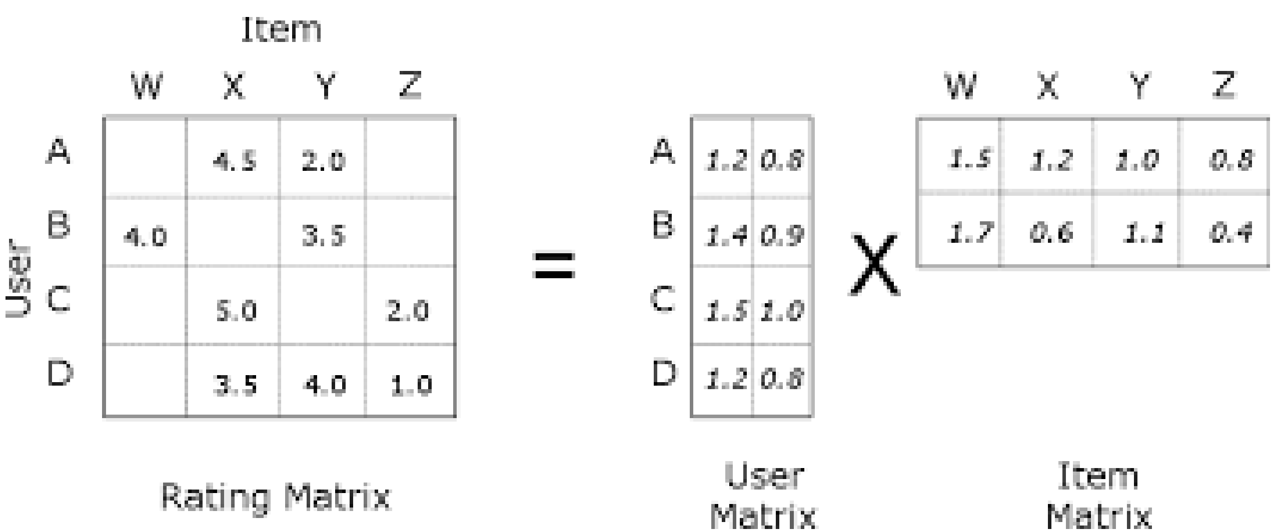
Learning Algorithm

- Stochastic gradient descent
- For each rating (u, i):
  - $update\ p_u: p_u \leftarrow p_u + \eta \cdot ((r_{ui} - \hat{r}_{ui})q_i - \lambda p_u)$
  - $update\ q_i: q_i \leftarrow q_i + \eta \cdot \left((r_{ui} - \hat{r}_{ui})p_u - \lambda q_i\right)$

  - Where η is the learning rate

Prediction Stage

- For an unseen pair (u, i)
  - $\hat{r}_{ui} = q_i^T p_u = p_u^T q_i$
- Example:
  - $r_{AW} = p_A^T q_W = 1.2 * 1.5 + 0.8 * 1.7 = 3.16$



Issues of CF

- **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.

Recommender Systems

- What is Recommender System?
- Collaborative Filtering
- Content-based Recommendation
- Evaluation Metrics
- Summary

Content-based recommendation

- Collaborative filtering does NOT require any information about the items,
  - However, it might be reasonable to exploit such information
  - E.g. recommend fantasy novels to people who liked fantasy novels in the past
- What do we need:
  - Some information about the available items such as the genre ("content")
  - Some sort of *user profile* describing what the user likes (the preferences)
- The task:
  - Learn user preferences
  - Locate/recommend items that are "similar" to the user preferences

Content representation and item similarities

User profile

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, Murder, Neo-nazism
...					

Item

Title	Genre	Author	Type	Price	Keywords
...	Fiction, Suspense	Brunonia Barry, Ken Follet, ..	Paperback	25.65	detective, murder, New York

- Simple approach
  - Compute the similarity of an unseen item with the user profile based on the keyword overlap (e.g. using the Dice coefficient)
  - $\mathrm{sim}(b_i, b_j) = \frac{2 * |keywords(b_i) \cap keywords(b_j)|}{|keywords(b_i)| + |keywords(b_j)|}$
- Other advanced similarity measure

Recommender Systems

- What is Recommender System?
- Collaborative Filtering
- Content-based Recommendation
- Evaluation Metrics
- Summary

Accuracy measures

- Datasets with items rated by users
  - MovieLens datasets 100K-10M ratings
  - Netflix 100M ratings
- Historic user ratings constitute ground truth
- Metrics measure error rate
  - Mean Absolute Error (*MAE*) computes the deviation between predicted ratings and actual ratings

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|p_i - r_i|$$

  - Root Mean Square Error (*RMSE*) is similar to *MAE*, but places more emphasis on larger deviation

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - r_i)^2}$$

Precision and Recall

- **Precision:** a measure of exactness, determines the fraction of relevant items retrieved out of all items retrieved
  - E.g. the proportion of recommended movies that are actually good

$$Precision = \frac{tp}{tp + fp} = \frac{|good\ movies\ recommended|}{|\text{all recommendations}|}$$

- **Recall:** a measure of completeness, determines the fraction of relevant items retrieved out of all relevant items
  - E.g. the proportion of all good movies recommended

$$Recall = \frac{tp}{tp + fn} = \frac{|good\ movies\ recommended|}{|all\ good\ movies|}$$

Recommender Systems

- What is Recommender System?
- Collaborative Filtering
- Content-based Recommendation
- Evaluation Metrics
- Summary

Summary

- Recommendation
  - User-based CF, matrix factorization-based CF
  - Content-based recommendation
  - Evaluation

References

- http://ijcai13.org/files/tutorial_slides/td3.pdf
- http://research.microsoft.com/pubs/115396/EvaluationMetrics.TR.pdf
- https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf