Collaborative Filtering

Big Data Analysis: Collaborative Filtering

Table of Content

Recommendation Systems

- Collaborative Filtering
  - Originally was published in WWW Conference

Item-Based Collaborative Filtering Recommendation Algorithms

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl
{sarwar, karypis, konstan, riedl}@cs.umn.edu
GroupLens Research Group/Army HPC Research Center
Department of Computer Science and Engineering
University of Minnesota, Minneapolis, MN 55455

ABSTRACT

Recommender systems apply knowledge discovery techniques to the problem of making personalized recommendations for information, products or services during a live interaction. These systems, especially the k-nearest neighbor collaborative filtering based ones, are achieving widespread success on the Web. The tremendous growth in the amount of available information and the number of visitors to Web sites in recent years poses some key challenges for recommender systems. These are: producing high quality recommendations, performing many recommendations per second for millions of users and items and achieving high coverage in the face of data sparsity. In traditional collaborative filtering systems the amount of work increases with the number of partici-

through all the available information to find that which is most valuable to us.

One of the most promising such technologies is *collaborative filtering* [19, 27, 14, 16]. Collaborative filtering works by building a database of preferences for items by users. A new user, Neo, is matched against the database to discover *neighbors*, which are other users who have historically had similar taste to Neo. Items that the neighbors like are then recommended to Neo, as he will probably also like them. Collaborative filtering has been very successful in both research and practice, and in both information filtering applications and E-commerce applications. However, there remain important research questions in overcoming two fundamental challenges for collaborative filtering recommender systems.

Recommendation Systems

- Collaborative Filtering
  - WWW is one of the most popular conference in the world

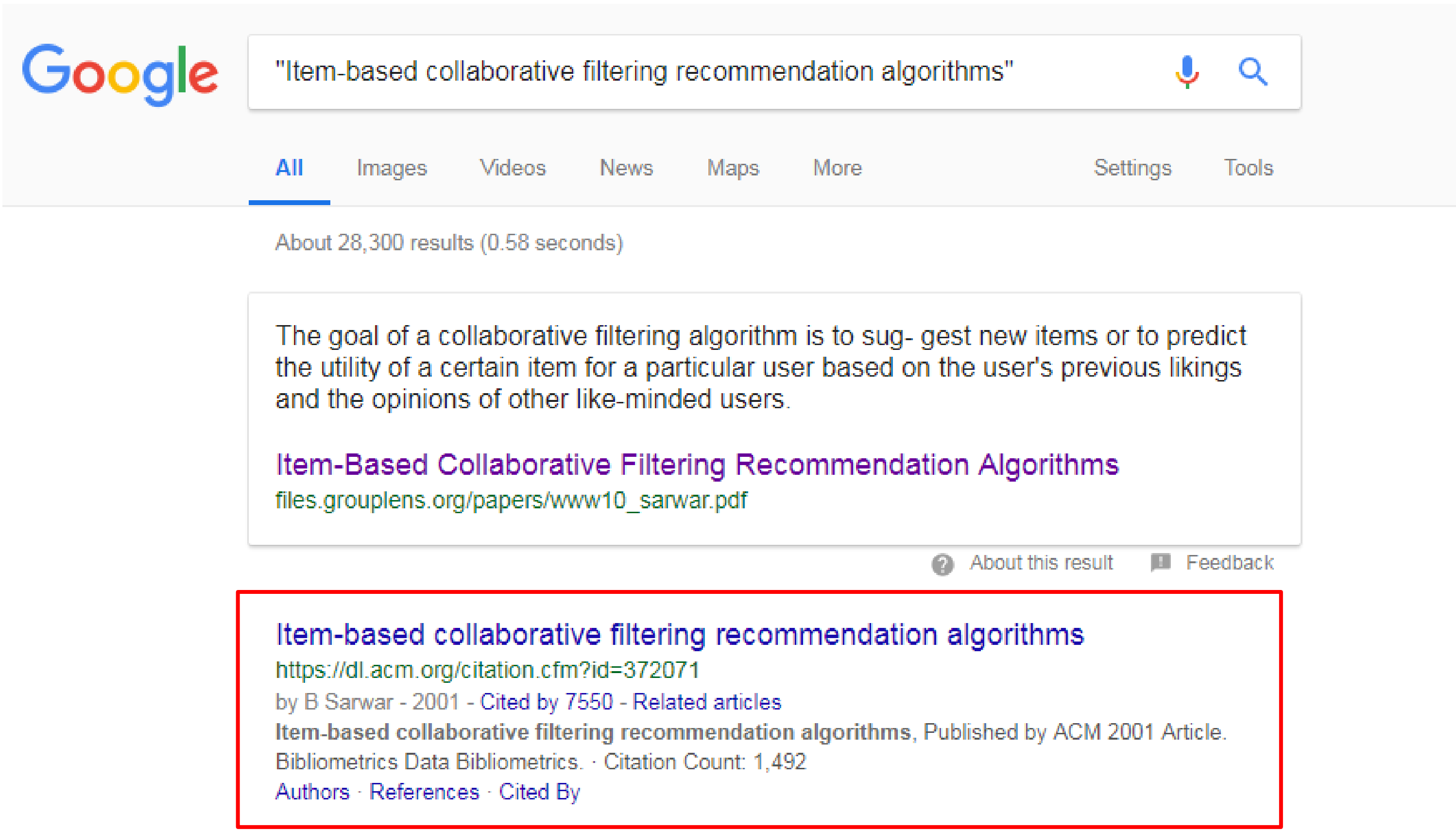
	Hindex	Publisher	Conference Details
1	158	IEEE	CVPR : IEEE Conference on Computer Vision and Pattern Recognition, CVPR Jun 15, 2019 - Jun 21, 2019 - Long Beach , **United States** http://cvpr2019.thecvf.com/
2	101	Neural Information Processing Systems Foundation	NIPS : Neural Information Processing Systems (NIPS) Dec 3, 2018 - Dec 6, 2018 - Palais des Congrès de Montréal , **Canada** https://nips.cc/
3	98	Springer	ECCV : European Conference on Computer Vision Sep 8, 2018 - Sep 14, 2018 - Munich , **Germany** https://eccv2018.org/
4	91	IMLS	ICML : International Conference on Machine Learning (ICML) Jul 10, 2018 - Jul 15, 2018 - Stockholm , **Sweden** https://icml.cc/
5	89	IEEE	ICCV : IEEE International Conference on Computer Vision Oct 27, 2019 - Nov 3, 2019 - Seoul , **South Korea** **Deadline : Mon 22 Apr 2019** http://iccv2019.thecvf.com/
6	85	acm Association for Computing Machinery	CHI : Computer Human Interaction (CHI) Apr 21, 2018 - Apr 21, 2018 - Montréal , **Canada** https://chi2018.acm.org/
7	80	IEEE	INFOCOM : Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) Apr 15, 2018 - Apr 19, 2018 - Honolulu HI , **United States** http://infocom2018.ieee-infocom.org/content/call-papers-main-conference
8	77	acm Association for Computing Machinery	WWW : International World Wide Web Conferences (WWW) May 13, 2019 - May 17, 2019 - San Francisco , **United States** https://www2019.thewebconf.org/

Recommendation Systems

- Collaborative Filtering
  - WWW is one of the most popular conference in the world

Recommendation Systems

- Collaborative Filtering
  - It became a grant success (Number of citations)



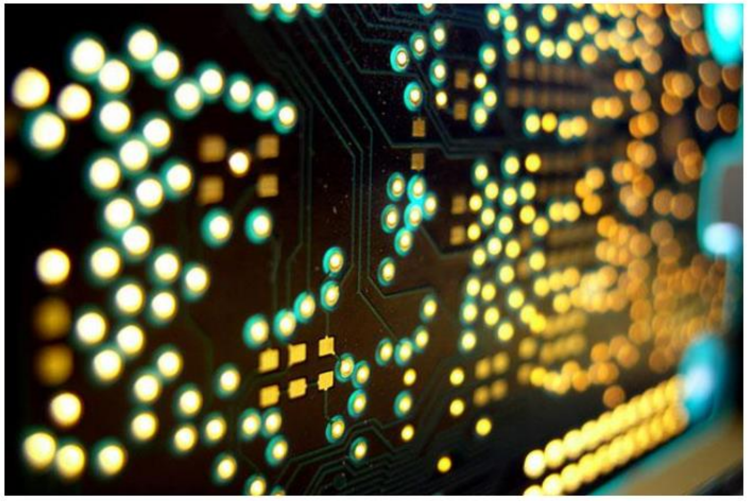
Recommendation Systems

- Collaborative Filtering
  - It became a grant success (UMN website)

University of Minnesota

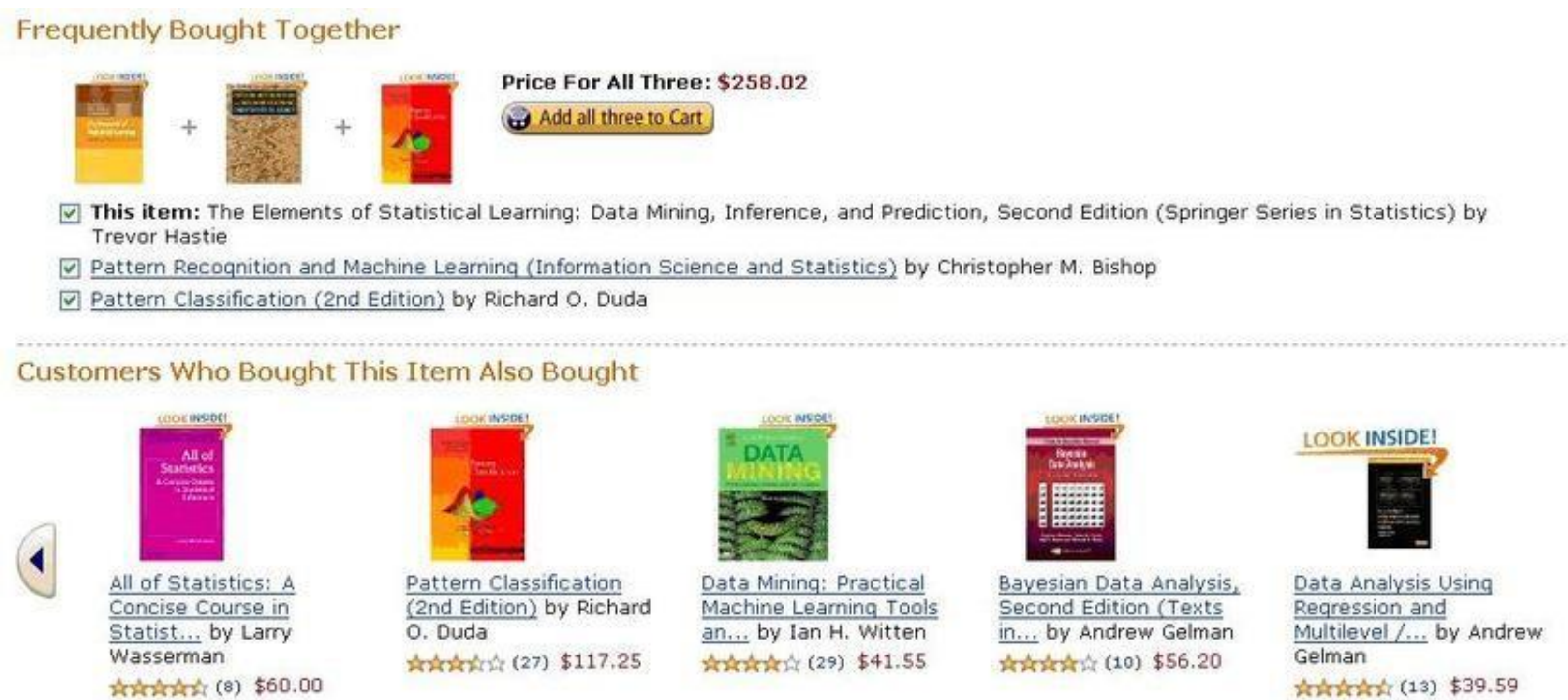
University of Minnesota professors and alumnus win international award for groundbreaking recommender systems research

March 24, 2016



Recommendation Systems

- Amazon recommendations

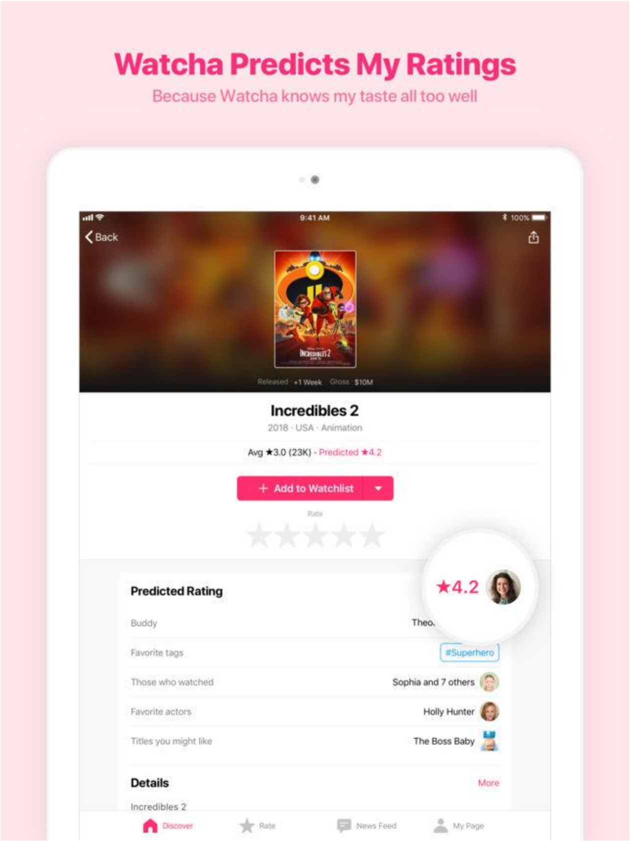


Recommendation Systems

- Netflix recommendations 옷 사

Recommendation Systems

- Watcha

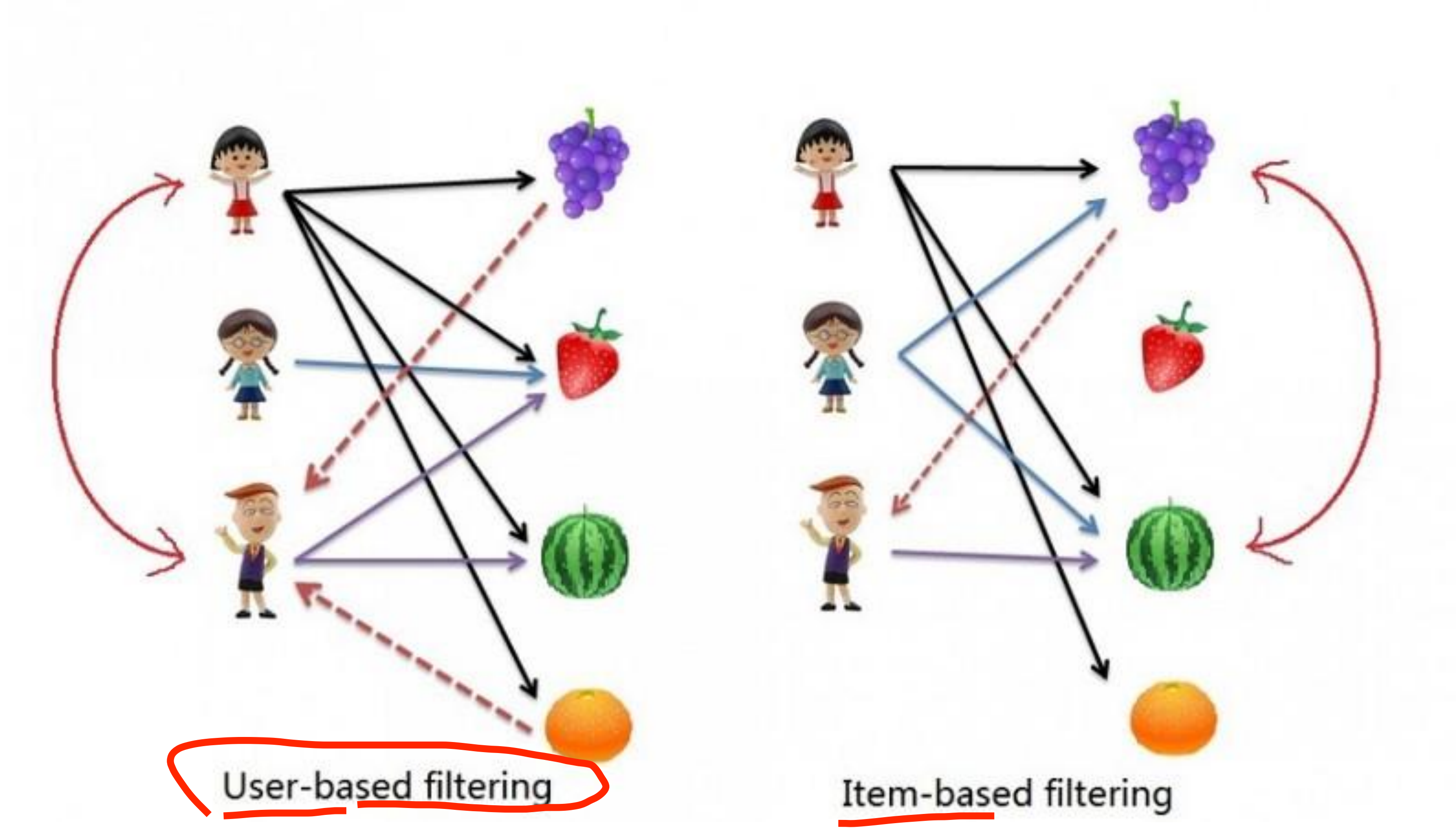


Recommendation Systems

- Summary
  - Amazon
    - 35% sales from recommendations
  - Google News
    - Recommendations generate 38% more clickthrough
  - Netflix
    - 2/3 of the movies watched are recommended
  - Spotify
    - 28% of the people would buy more music if they found what they liked

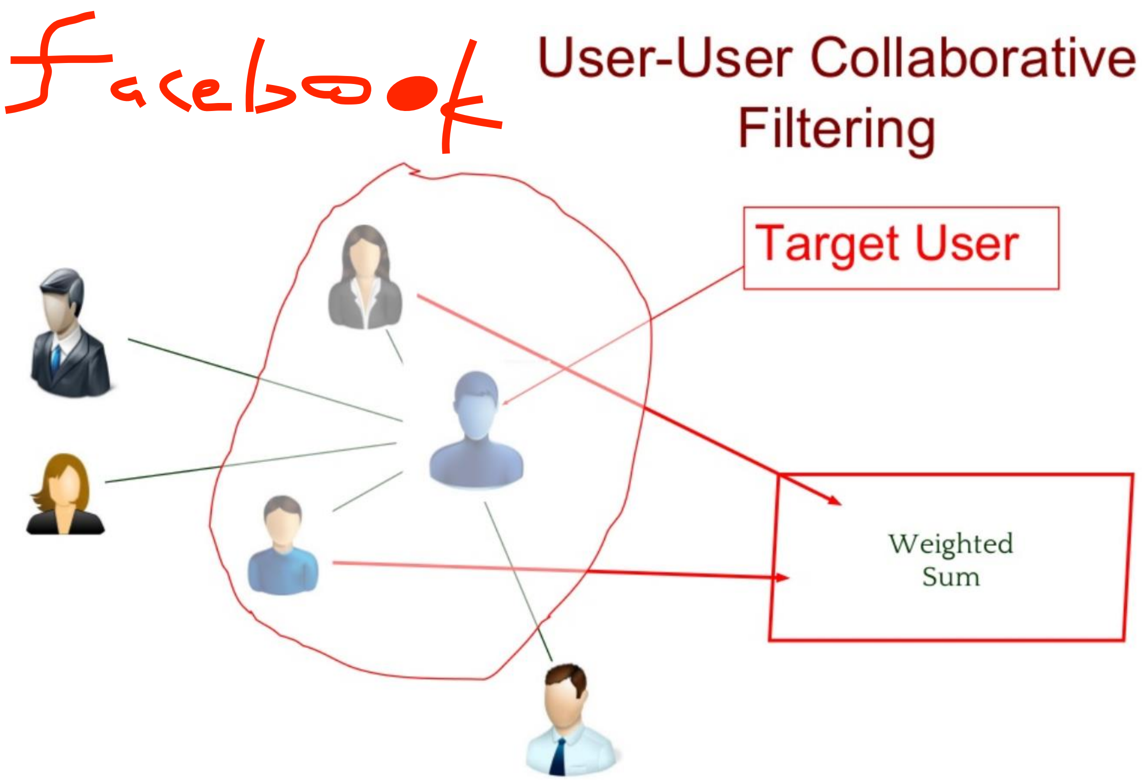
Collaborative Filtering (CF)

- Two types of CF



User-based Collaborative Filtering

- The basic idea
  - Finding the similar users who have the same interests as targer user



User-based Collaborative Filtering

- Example
  - A database of ratings of the current user, Alice, and some other users is given:

	Titanic	Avengers	Intern	Sin City	Parasite
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

  - Determine whether Alice will like or dislike **Parasite**, which Alice has not yet rated or seen

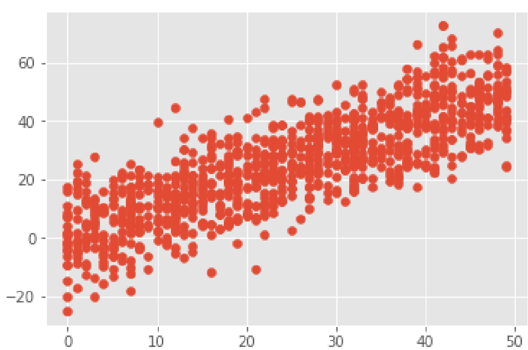
User-based Collaborative Filtering

- Some first questions
  - How do we measure similarity?
  - How many neighbors should we consider?
  - How do we generate a prediction from the neighbors' ratings?

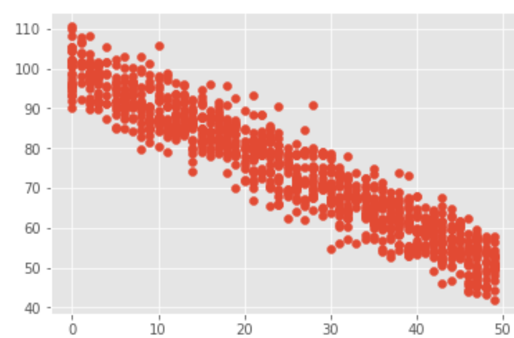
	Titanic	Avengers	Intern	Sin City	Parasite
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

User-based Collaborative Filtering

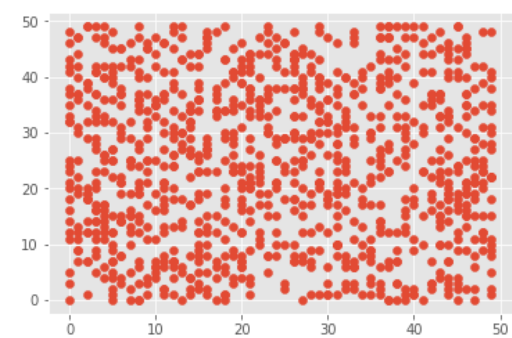
- Pearson Correlation
  - Possible similarity values between -1 and 1
    - Positive relationship when close to 1
    - Negative relationship when close to – 1
    - No relationship when 0



<Positive>



<Negative>



<No Corr>

User-based Collaborative Filtering

- Pearson Correlation
  - A popular similarity measure in user-based CF
    - a, b : users
    - $r_{a,p}$: rating of user a for item p
    - $\bar{r}$: mean user rating
    - P: a set of items, rated both by a and b

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

User-based Collaborative Filtering

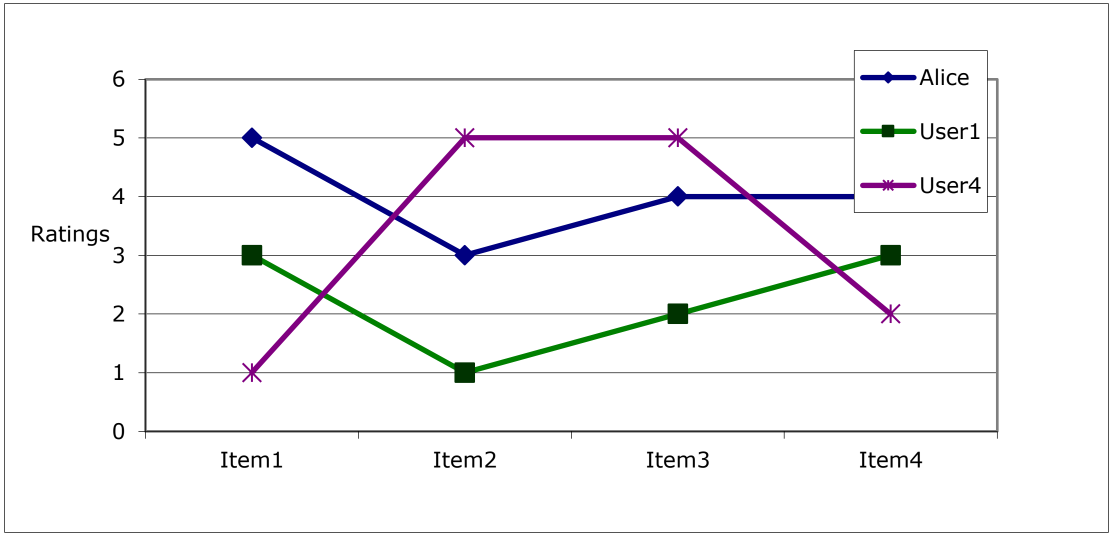
- Pearson Correlation
  - A popular similiraty measure in user-based Collaborative Filtering
    - *a*, *b* : users
    - $r_{a,p}$: rating of user *a* for item *p*
    - $\bar{r}$: mean user rating
    - *P*: a set of items, rated both by *a* and *b*

	Titanic	Avengers	Intern	Sin City	Parasite
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 = 0.7
sim = 0.00
sim = -0.79

User-based Collaborative Filtering

- Pearson Correlation
  - Takes differences in rating behavior into account



- Works well in usual domains, compared with alternative measures
  - such as cosine similarity

User-based Collaborative Filtering

- Pearson Correlation (two attributes) in Python
  - Numpy library and pearsonr function are used

```
pip install numpy

import numpy as np

from scipy.stats import pearsonr

alice = np.array([5, 3, 4, 4])
user1 = np.array([3, 1, 2, 3])
user2 = np.array([4, 3, 4, 3])
user3 = np.array([3, 3, 1, 5])
user4 = np.array([1, 5, 5, 2])

corr = pearsonr(alice, user1)

print(corr)
```

User-based Collaborative Filtering

- Pearson Correlation (multiple attributes) in Python
  - Numpy library and corrcoef function are used

```
import numpy as np

from scipy.stats import pearsonr

matrix = np.array([[5, 3, 4, 4],
                   [3, 1, 2, 3],
                   [4, 3, 4, 3],
                   [3, 3, 1, 5],
                   [1, 5, 5, 2]
                  ])

corr = np.corrcoef(matrix)

print(corr)
```

User-based Collaborative Filtering

- Pearson Correlation in Python
  - Visualization using matplotlib

```
pip install matplotlib

import matplotlib
import matplotlib.pyplot as plt
matplotlib.style.use('ggplot')

alice = np.array([5, 3, 4, 4])
user1 = np.array([3, 1, 2, 3])
user2 = np.array([4, 3, 4, 3])
user3 = np.array([3, 3, 1, 5])
user4 = np.array([1, 5, 5, 2])

plt.scatter(alice, user1)
plt.show()
```

User-based Collaborative Filtering

- Limitations
  - Systems performed poorly
    - When they had many items but comparatively few ratings
  - Computing similarities between all pairs of users was expensive
  - User profiles changed quickly and the entire system model had to be recomputed

Item-based collaborative filtering

- The basic idea

Item-based collaborative filtering

- Basic idea:
  - Use the similarity between items (and not users) to make predictions
- Example:
  - Look for items that are similar to Item5
  - Take Alice's ratings for these items to predict the rating for Item5

	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

Item-based collaborative filtering

- **Produces better results in item-to-item filtering**
- **Ratings are seen as vector in n-dimensional space**
- **Similarity is calculated based on the angle between the vectors**

$$similarity = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

- **Adjusted cosine similarity**
  - take average user ratings into account, transform the original ratings
  - U: set of users who have rated both items a and b

$$sim(\vec{a},\vec{b}) = \frac{\sum_{u \in U}(r_{u,a} - \overline{r_u})(r_{u,b} - \overline{r_u})}{\sqrt{\sum_{u \in U}(r_{u,a} - \overline{r_u})^2} \sqrt{\sum_{u \in U}(r_{u,b} - \overline{r_u})^2}}$$

Item-based collaborative filtering

- **Produces better results in item-to-item filtering**
- **Ratings are seen as vector in n-dimensional space**
- **Similarity is calculated based on the angle between the vectors**

$$similarity = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

	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
	0.99	0.73	0.72	0.93	

Item-based collaborative filtering

- Cosine Similarity (two attributes) in Python
  - Sklearn library and cosine_similarity function are used

```
pip install sklearn

from sklearn.metrics.pairwise import
cosine_similarity

cosine_similarity([[3, 4, 3, 1]], [[3, 5, 4, 1]])

cosine_similarity([[3, 3, 5, 2]], [[3, 5, 4, 1]])

cosine_similarity([[1, 3, 3, 5]], [[3, 5, 4, 1]])

cosine_similarity([[2, 4, 1, 5]], [[3, 5, 4, 1]])
```

Summary and Discussions

- Collaborative Filtering
  - User-Based CF
  - Item-Based CF
- Similarity Measures
  - Pearson Correlation
  - Cosine Similarity
- Limitations
- Python implementation of CF

Homework for Lecture 14

- Submit your source code for the following task:
  1. Try all source code in the lecture

- Submission: source code, result screenshots and result ex planation

Q&A

This lecture is supported by Seondo project of the Ministry of Education in Korea.