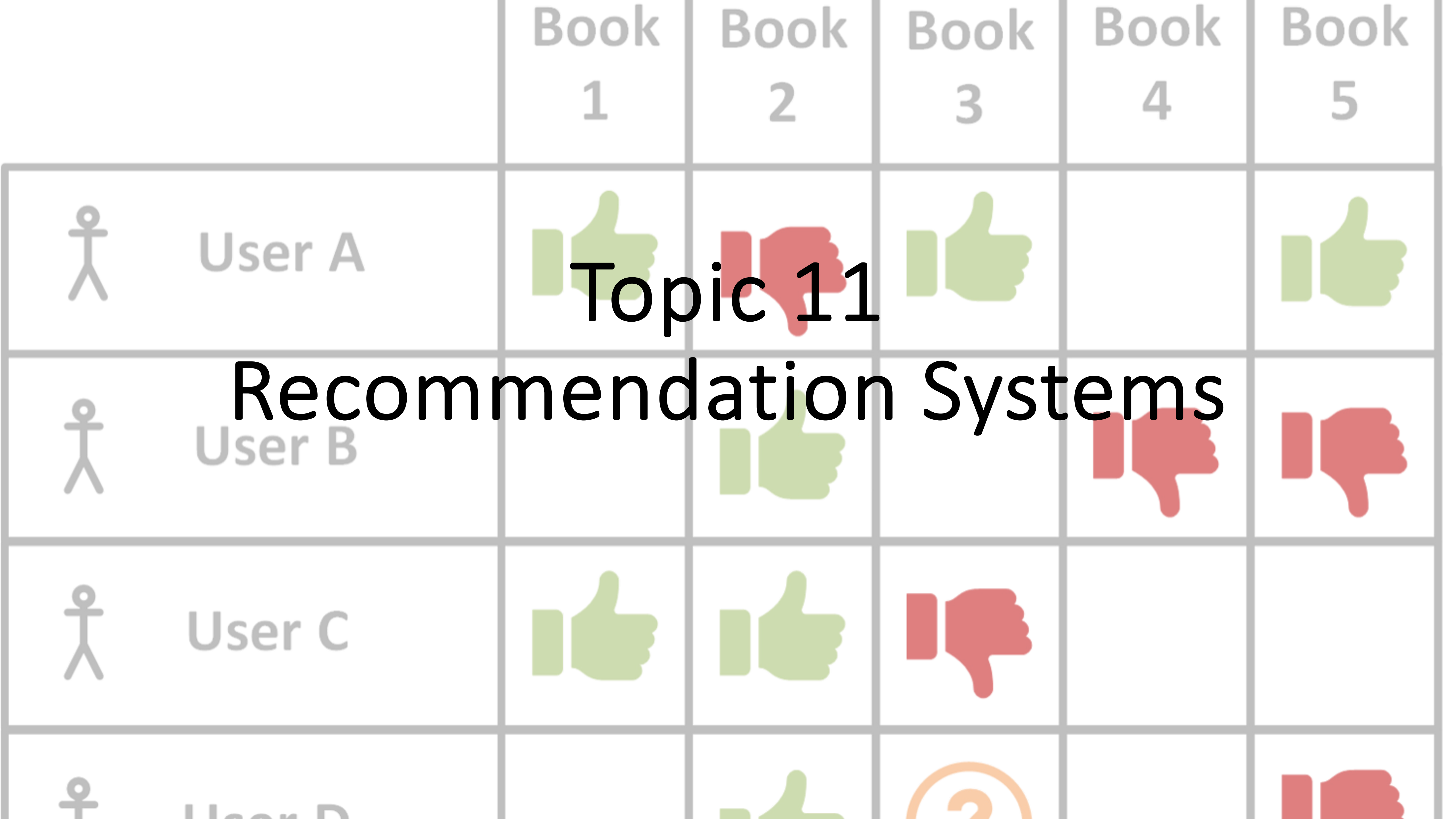
Topic 11
Recommendation Systems



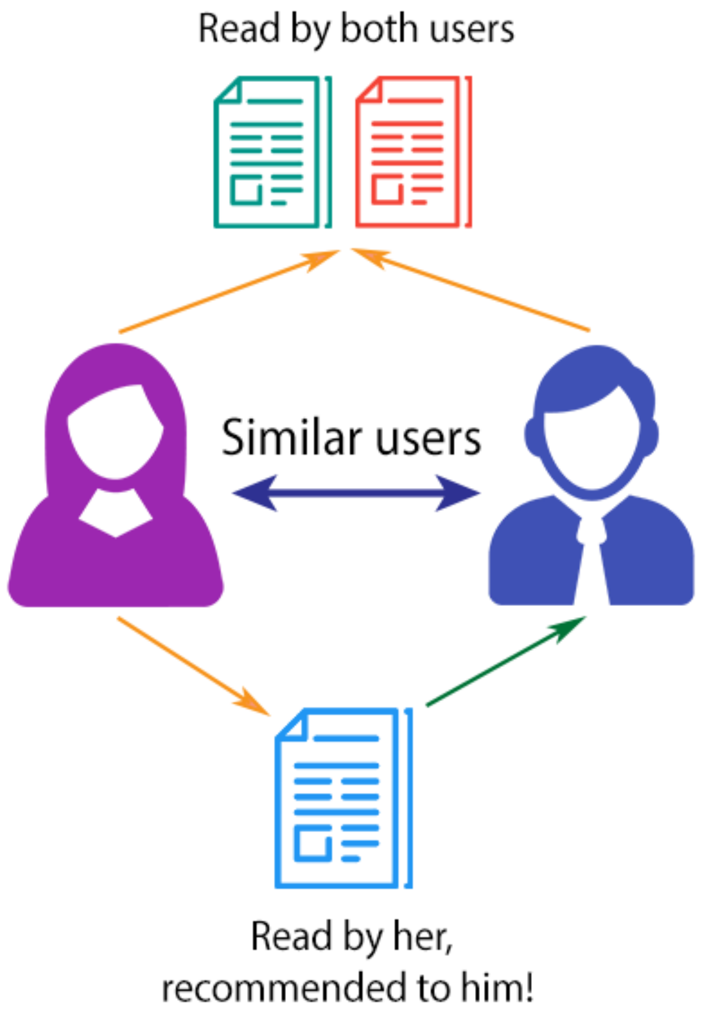
A Recommendation System…

- Is a way to filter information
- Deals with choice overload
- Is focused on customer preference, interest, and observed behavior

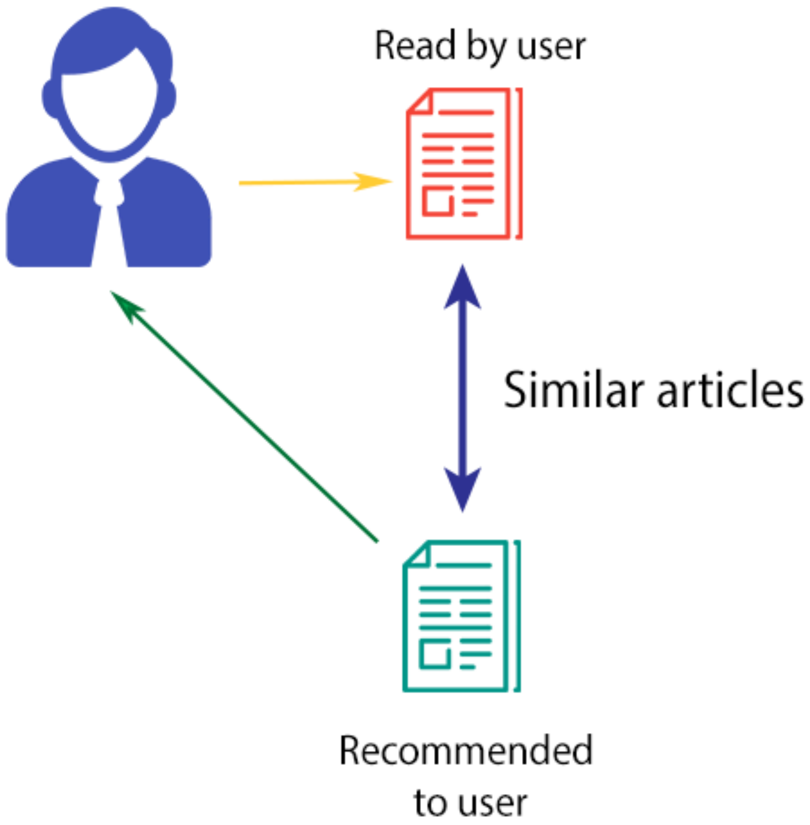
All sorts of websites use recommendation systems

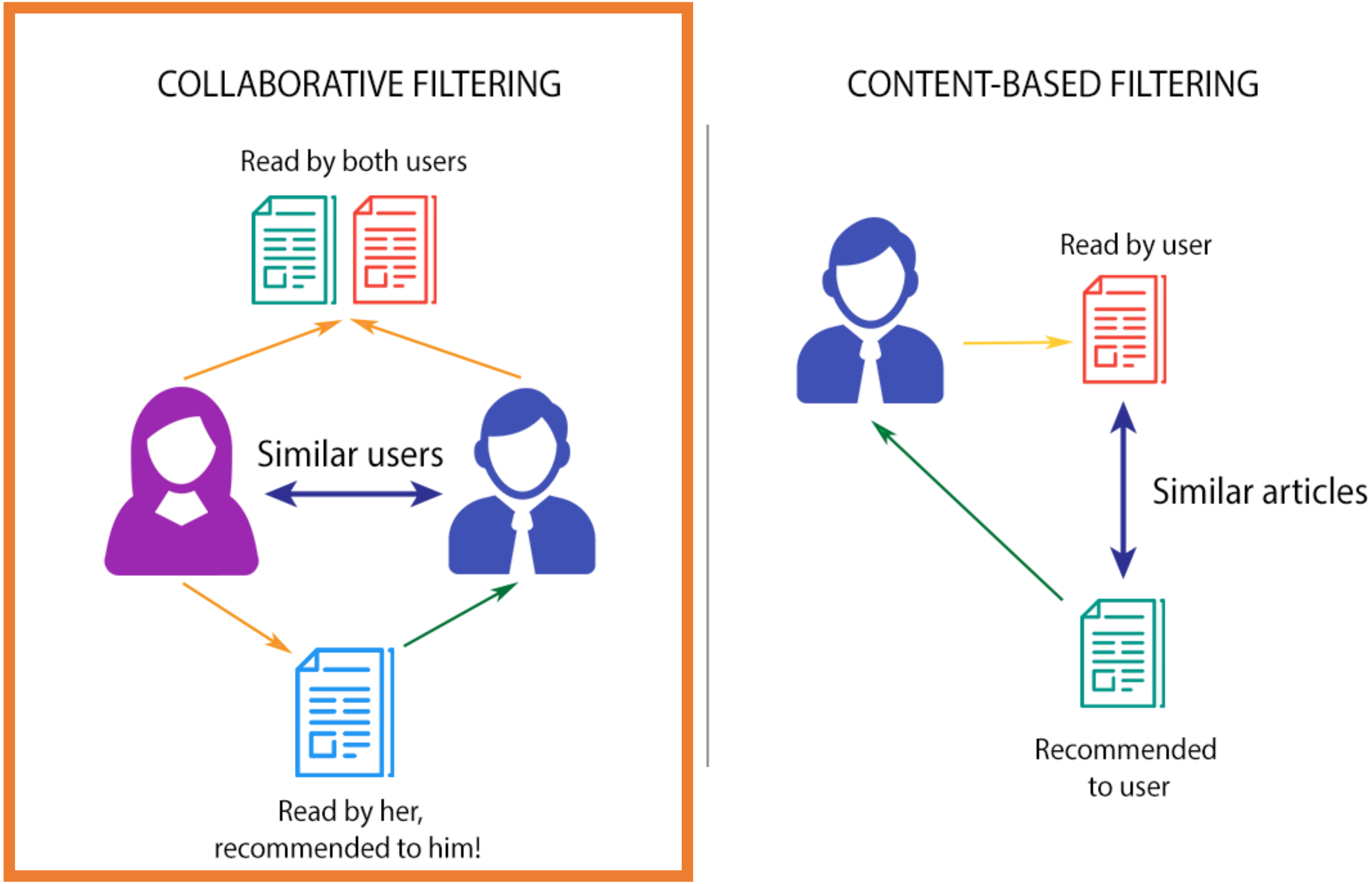
- Facebook
- Netflix
- Linkedin
- Amazon
- Youtube
- Pinterest

COLLABORATIVE FILTERING



CONTENT-BASED FILTERING



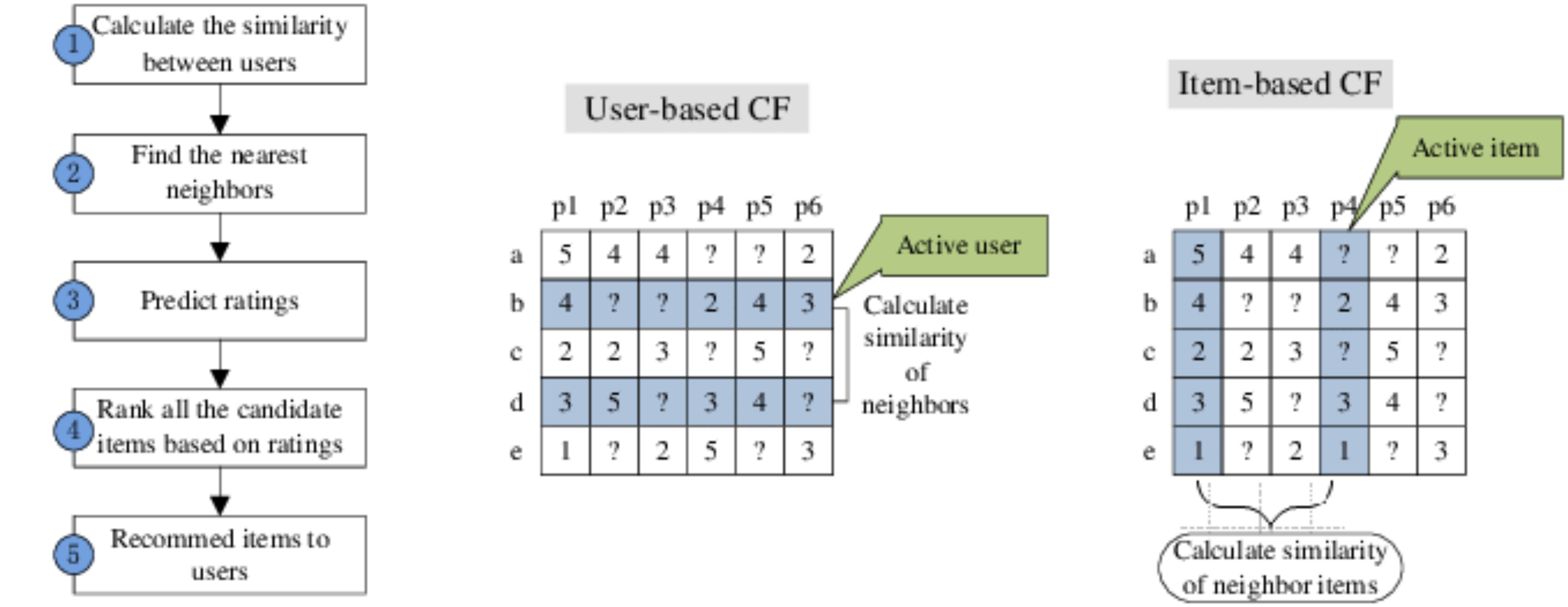

COLLABORATIVE FILTERING
Read by both users
Similar users
Read by her, recommended to him!
CONTENT-BASED FILTERING
Read by user
Similar articles
Recommended to user

Types of Collaborative Filtering

- User-based
  - Try to search for lookalike customers and offer products based on what they have chosen
- Item-based
  - Look for similar items based on user preferences

- “Ratings” can be implicit or explicit
  - Real-valued matrix: The interaction is well quantified, such as ratings, number of visits…
  - Binary matrix: The interaction is a binary preference, such as like/dislike.
  - One-class matrix: The case of implicit feedback–only positive reactions are recorded.

User-based vs Item-based



User-based CF

	p1	p2	p3	p4	p5	p6
a	5	4	4	?	?	2
b	4	?	?	2	4	3
c	2	2	3	?	5	?
d	3	5	?	3	4	?
e	1	?	2	5	?	3

Item-based CF

	p1	p2	p3	p4	p5	p6
a	5	4	4	?	?	2
b	4	?	?	2	4	3
c	2	2	3	?	5	?
d	3	5	?	3	4	?
e	1	?	2	1	?	3

The procedure of memory-based collaborative filtering RS

https://medium.com/analytics-vidhya/matrix-factorization-made-easy-recommender-systems-7e4f50504477

Matrix factorization

Item

User	W	X	Y	Z
A		4.5	2.0	
B	4.0		3.5	
C		5.0		2.0
D		3.5	4.0	1.0

Rating Matrix

=

A	*1.2*	*0.8*
B	*1.4*	*0.9*
C	*1.5*	*1.0*
D	*1.2*	*0.8*

User Matrix

X

W	X	Y	Z
1.5	*1.2*	*1.0*	*0.8*
1.7	*0.6*	*1.1*	*0.4*

Item Matrix

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

Matrix factorization

Item

User	W	X	Y	Z
A		4.5	2.0	
B	4.0		3.5	
C		5.0		2.0
D		3.5	4.0	1.0

Rating Matrix

=

A	*1.2*	*0.8*
B	*1.4*	*0.9*
C	*1.5*	*1.0*
D	*1.2*	*0.8*

User Matrix

X

W	X	Y	Z
1.5	*1.2*	*1.0*	*0.8*
1.7	*0.6*	*1.1*	*0.4*

Item Matrix

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

Matrix factorization

Rating Matrix (User × Item)

User	W	X	Y	Z
A		4.5	2.0	
B	4.0		3.5	
C		5.0		2.0
D		3.5	4.0	1.0

=

User Matrix

A	1.2	0.8
B	1.4	0.9
C	1.5	1.0
D	1.2	0.8

X

Item Matrix

W	X	Y	Z
1.5	1.2	1.0	0.8
1.7	0.6	1.1	0.4

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

Matrix factorization

$$
\begin{array}{cc}
 & \text{Item} \\
\text{User} &
\begin{array}{c|cccc}
 & W & X & Y & Z \\
\hline
A & & 4.5 & 2.0 & \\
B & 4.0 & & 3.5 & \\
C & & 5.0 & & 2.0 \\
D & & 3.5 & 4.0 & 1.0
\end{array}
\end{array}
=
\begin{array}{c|cc}
A & 1.2 & 0.8 \\
B & 1.4 & 0.9 \\
C & 1.5 & 1.0 \\
D & 1.2 & 0.8
\end{array}
\times
\begin{array}{cccc}
W & X & Y & Z \\
\hline
1.5 & 1.2 & 1.0 & 0.8 \\
1.7 & 0.6 & 1.1 & 0.4
\end{array}
$$

$$
\min_{p\cdot,q\cdot,b\cdot} \sum_{(u,i)\in K} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda (\| p_u \|^2 + \| q_i \|^2 + b_u^2 + b_i^2)
$$

Example

```
from scipy.sparse import csr_matrix
adj_matrix = csr_matrix((ratings, (user_data, item_data)))
```

Items

Users	1	2	...	i	...	m
1	5	3		1	2	
2		2				4
:			5			
u	3	4		2	1	
:					4	
n			3	2		

```
from sklearn.utils.extmath import randomized_svd
import numpy as np

U, S, VT = randomized_svd(adj_matrix, n_components=5,n_iter=5, random_state=None)
```

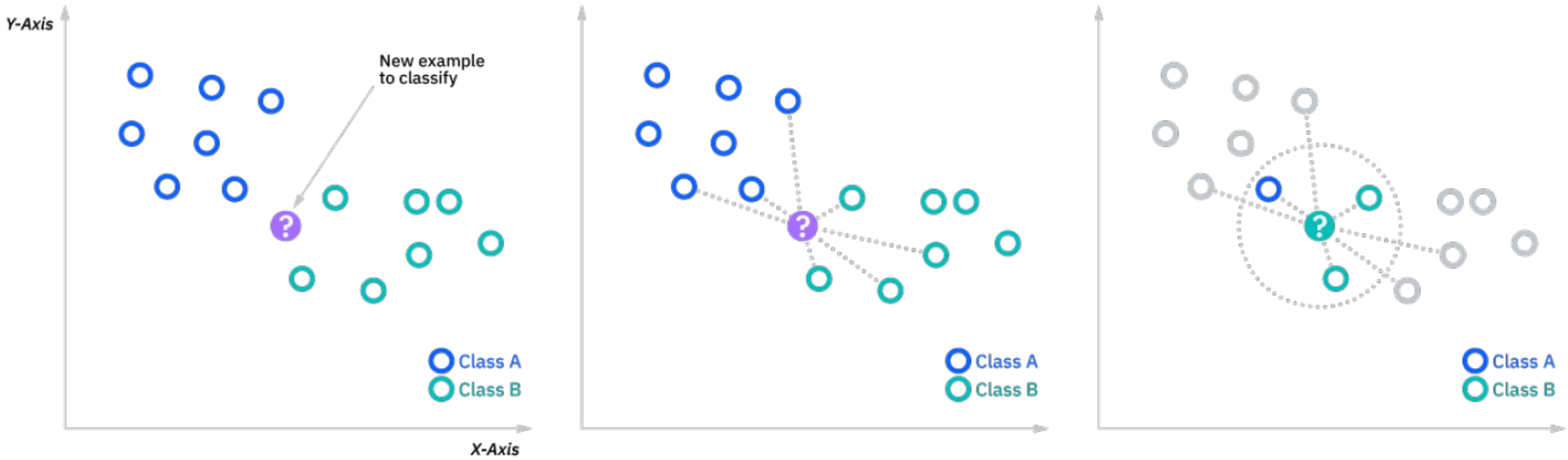
```
predicted_rating(u, i) = U[u,:] dot (S * VT[:,i])
```

Example

```
import implicit

# initialize a model
model = implicit.als.AlternatingLeastSquares(factors=50)

# train the model on a sparse matrix of item/user/confidence weights
model.fit(item_user_data)

# recommend items for a user
user_items = item_user_data.T.tocsr() recommendations = model.recommend(userid, user_items)

# find related items
related = model.similar_items(itemid)
```

Options for similarities for users or content

- K-Nearest Neighbors
  - KNN is a machine learning algorithm to find clusters of similar users based on common ratings
  - We find the k items that have the most similar user engagement vector



Pearson Correlation

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	4	5	-	-	4	-
User 2	-	-	4	5	-	-
User 3	-	3	-	4	5	4
User 4	3	-	5	-	-	-
User 5	-	4	-	-	-	5

Correlation Coefficient Formula

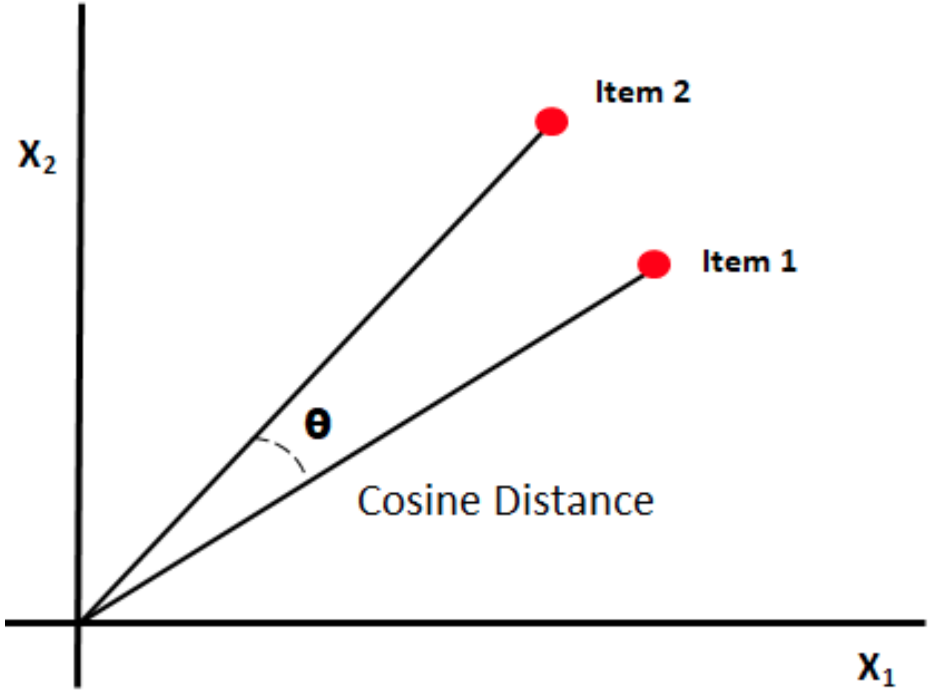
$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{\left[n\Sigma x^2 - (\Sigma x)^2\right]\left[n\Sigma y^2 - (\Sigma y)^2\right]}}$$

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
Movie 1	1	0.41	0.41	-0.98	-0.05	0.87
Movie 2	0.41	1	-0.42	0.44	0.56	-0.05
Movie 3	0.41	-0.42	1	0.56	0.87	-0.05
Movie 4	-0.98	0.44	0.56	1	0.05	-0.87
Movie 5	-0.05	0.56	0.87	0.05	1	0.41
Movie 6	0.87	-0.05	-0.05	-0.87	0.41	1

If User 1 has liked Movie 1 and Movie 5, we can recommend Movie 6, which has a high similarity score with both of those movies.

Types of Similarity between Embeddings

- Euclidean
- **Cosine**
- Manhattan
- etc.



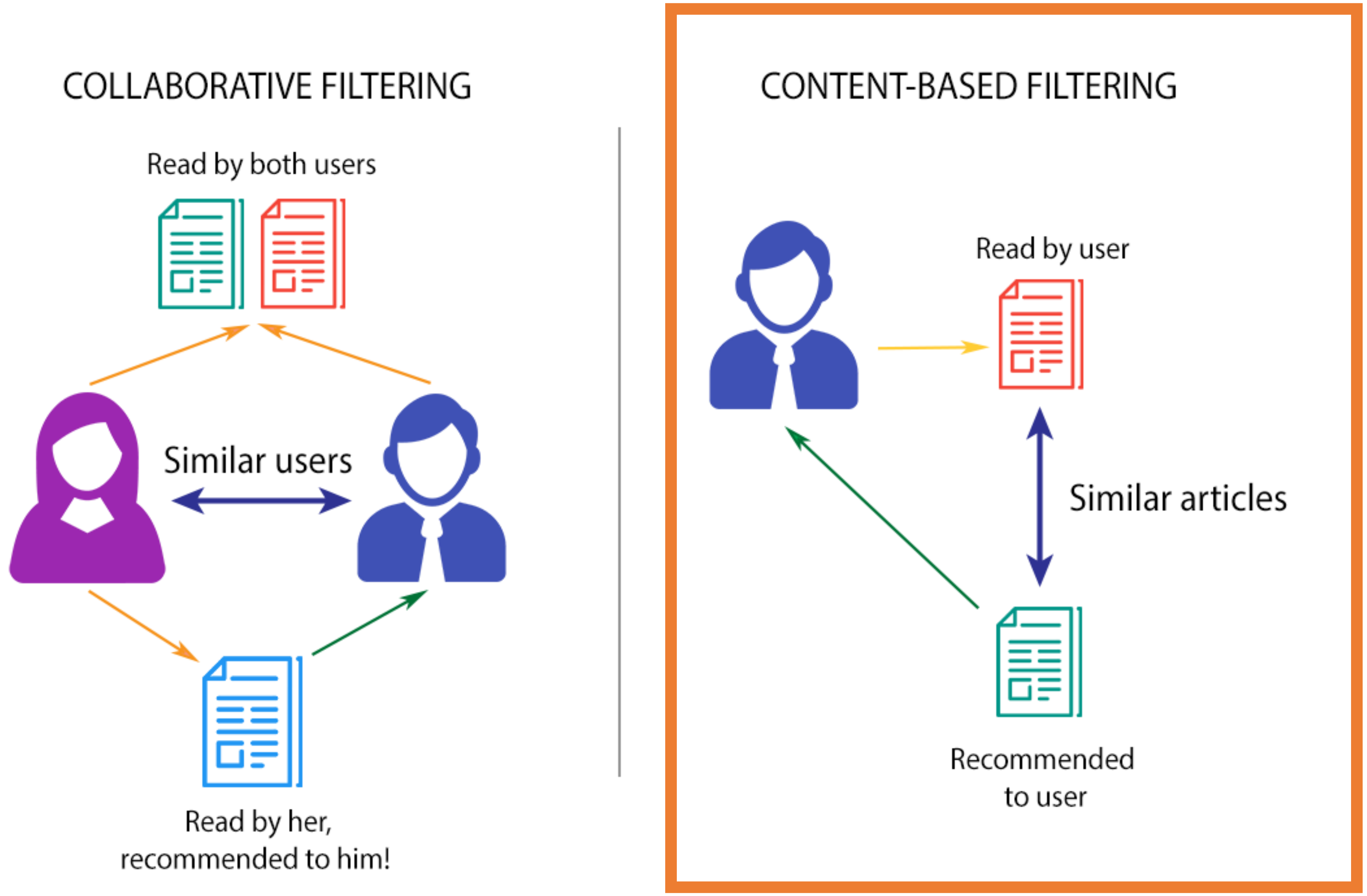
$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Drawbacks

- Data sparsity
- Nearest neighbours doesn’t scale well
- May end up defaulting to popular items
- New user cold start problems
- New item cold start problems

Drawback for us?

- Usually do not require NLP-based solutions


COLLABORATIVE FILTERING
Read by both users
Similar users
Read by her,
recommended to him!
CONTENT-BASED FILTERING
Read by user
Similar articles
Recommended
to user

Book Recommender Example

- https://github.com/practical-nlp/practical-nlp-code/blob/master/Ch7/04_RecommenderSystems.ipynb

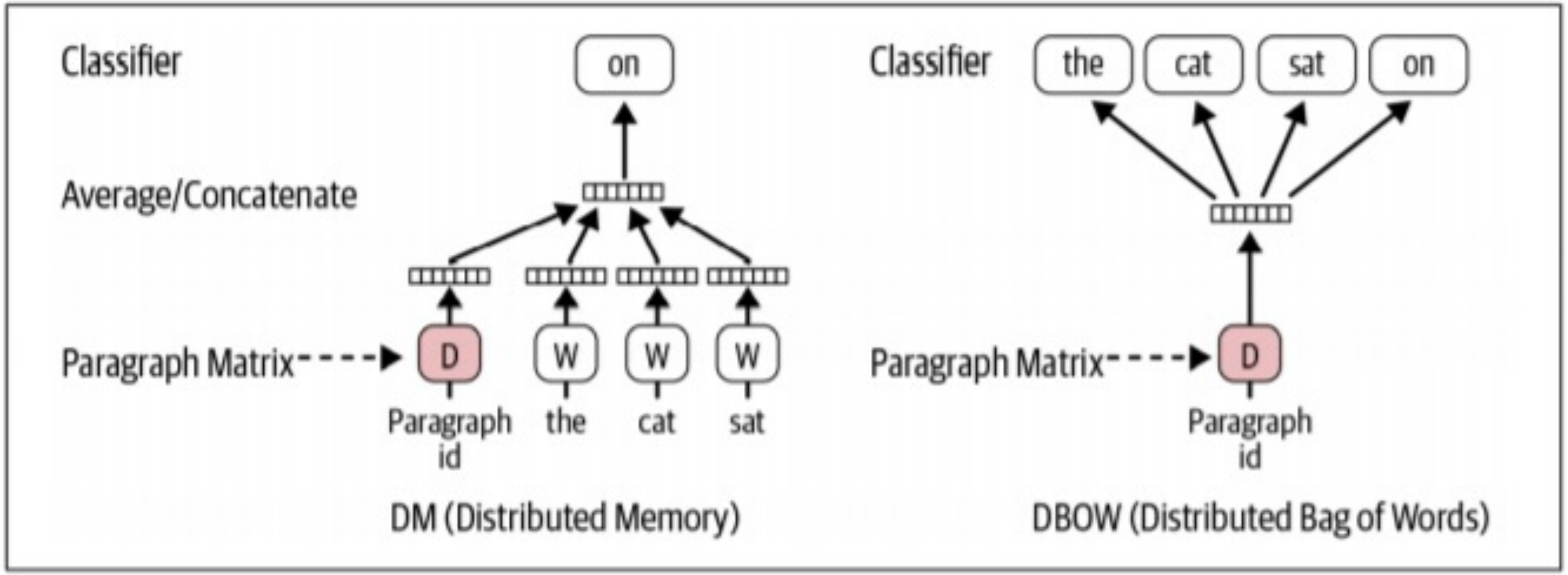


Figure 3-13. Doc2vec architectures: (a) DM and (b) DBOW

Advantages

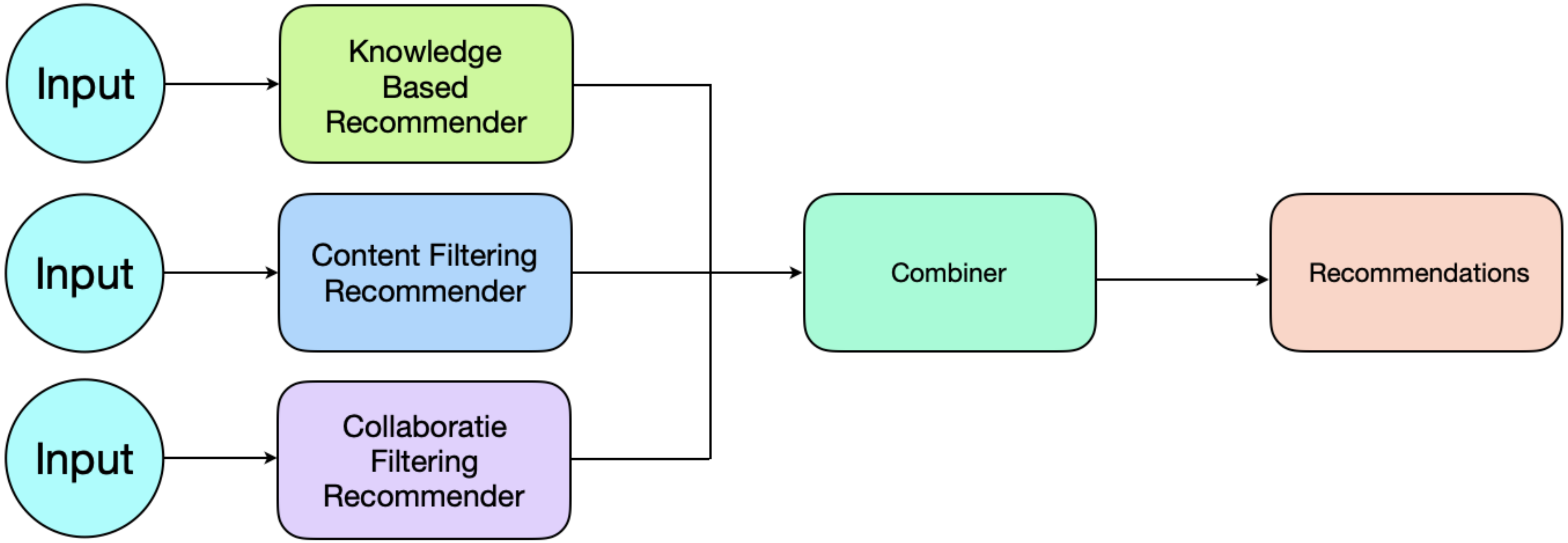
- Particularly useful when there is a large number of items and limited user data available.
  - Remove cold start and data about other users
  - If user has unique taste this method will work
  - Can recommend new and unpopular items
- Can provide personalized recommendations based on specific user preferences, as it focuses on the features that the user has previously liked.
  - Doesn’t need evaluations from other users

Disadvantages

- Can result in recommendations that are too similar to items that the user has already seen or interacted with
- May not capture the complexity of user preferences
  - Focused on specific item attributes rather than overall user behavior or patterns

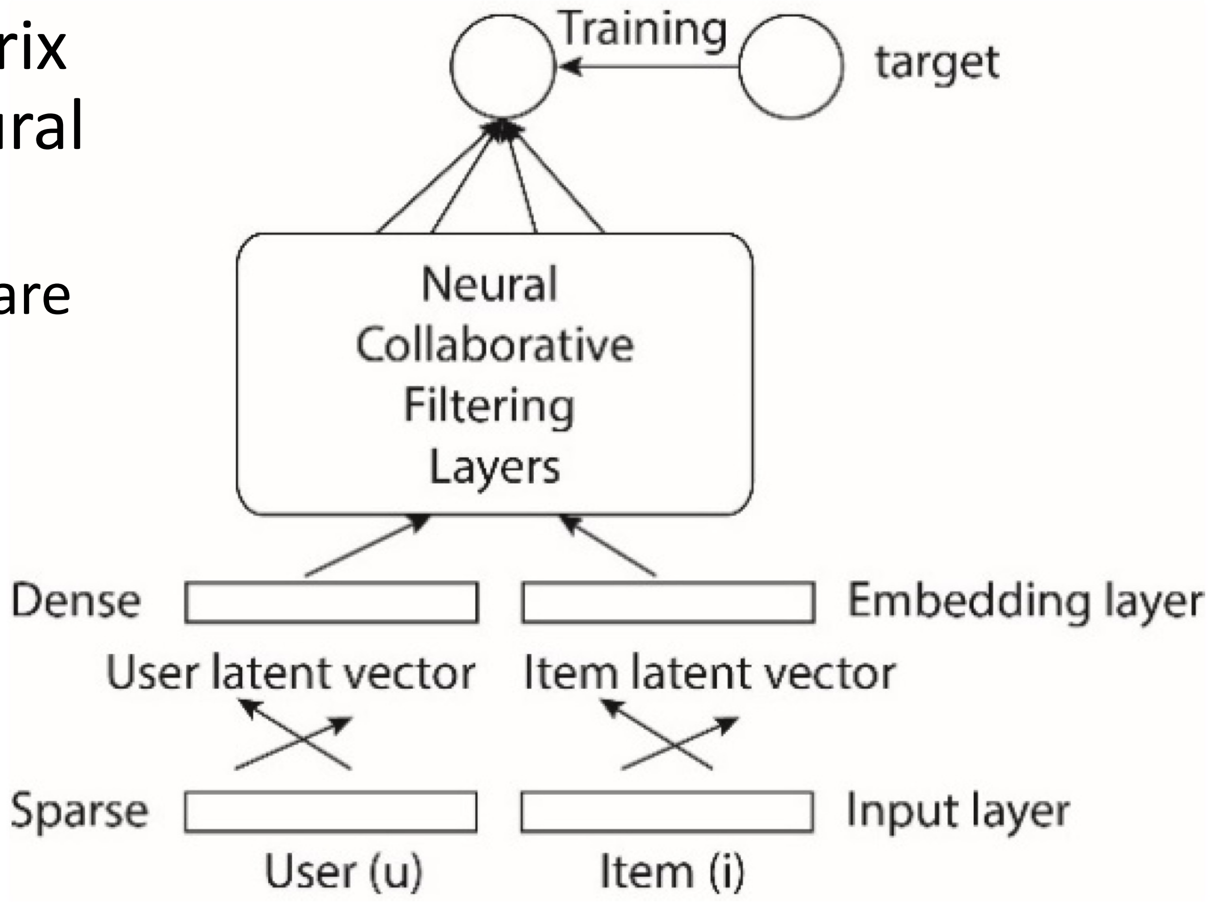
Hybrid Solutions

- Make recommendations by comparing the ratings, watching, and searching habits of similar users (i.e. collaborative filtering) as well as items that share characteristics with other items a user has rated highly (content-based filtering).



Neural collaborative filtering (NCF) model

- Hybrid model that combines matrix factorization techniques with neural networks
  - User and item embeddings, which are learned through a combination of matrix factorization and neural network training



Activity

- Building a Recommendation System
- Pick one of the following recommendation systems to finish implementing in the exercise notebook
  - One option is to make a recommendation system based on what songs users have in their playlists
  - The other option is to finish making the recommendation system based on song lyrics similarity

Final questions:

1) **Given that a hypothetical user has a playlist of 10 songs, recommend 10 other songs that the user has not listened to before that they might want to add to their playlist.**
2) **Brainstorm weaknesses with the current set up or dataset being used.**

Next time

- Text generation