JOHANNES KEPLER UNIVERSITÄT LINZ

Institute of Computational Perception

MULTIMEDIA MINING AND SEARCH GROUP

UE Learning from User-generated Data S2022
Session 2: Collaborative Filtering

Oleg Lesota (**oleg.lesota@jku.at**)

Agenda

- Exercise 1: Hints & Solutions
- Collaborative Filtering
- Exercise 2: CF on implicit feedback

Exercise 1: Interaction Matrix

IN:

usr_path = *'sampled_1000_items_demo.txt'*,

itm_path = *'sampled_1000_items_tracks.txt'*,

inter_path = *'sampled_1000_items_inter.txt'*,

threshold = **1**

OUT:

interaction_matrix

Exercise 1: Interaction Matrix

1. Get dimensions:

'sampled_1000_items_tracks.txt

Artist	Track Name
Helstar	Harsh Reality
Carpathian Forest	Dypfryst / Dette Er Mitt Helvete
Cantique Lépreux	Tourments Des Limbes Glacials
...	...

'sampled_1000_items_demo.txt

Location	Age	Gender	Reg. Date
BR	25	m	2007-10-12 18:42:00
UK	27	m	2006-11-17 16:51:56
US	32	m	2010-02-02 22:30:15
...	...	...	...

$$\begin{array}{cc} & \text{\# of items} \\ \text{\# of users} & \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \end{bmatrix} \end{array}$$

Exercise 1: Interaction Matrix

1. Get dimensions:

'sampled_1000_items_tracks.txt

Artist	Track Name
Helstar	Harsh Reality
Carpathian Forest	Dypfryst / Dette Er Mitt Helvete
Cantique Lépreux	Tourments Des Limbes Glacials
...	...

'sampled_1000_items_demo.txt

Location	Age	Gender	Reg. Date
BR	25	m	2007-10-12 18:42:00
UK	27	m	2006-11-17 16:51:56
US	32	m	2010-02-02 22:30:15
...	...	...	...

$$\begin{array}{cc} & \text{\# of items} \\ \text{\# of users} & \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \end{bmatrix} \end{array}$$

2. Conditionally Populate the matrix:

User Id	Track Id	Number of Interactions
0	1	3
0	6	5
2	17	8
...	...	...

3 >= threshold?

yes

$$\begin{bmatrix} 0 & 1 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

Exercise 1: POP Recommender

IN:

```
inter_matr,   # interaction matrix
user,         # user ID
top_k,        # number of recommendations
```

OUT:

```
top_pop       # array of IDs of recommended items
```

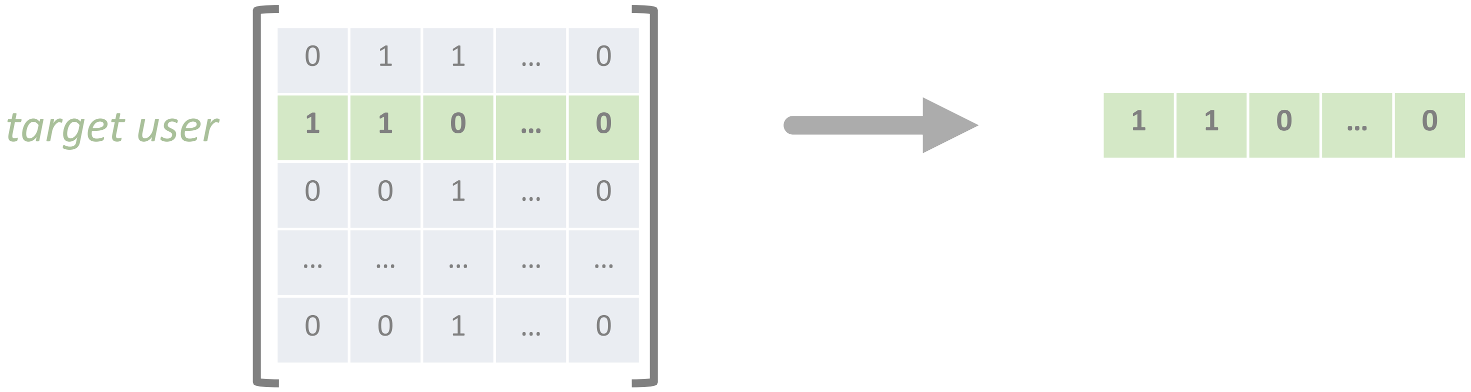
Exercise 1: POP Recommender

1. Save IDs of items seen by the target ***user***

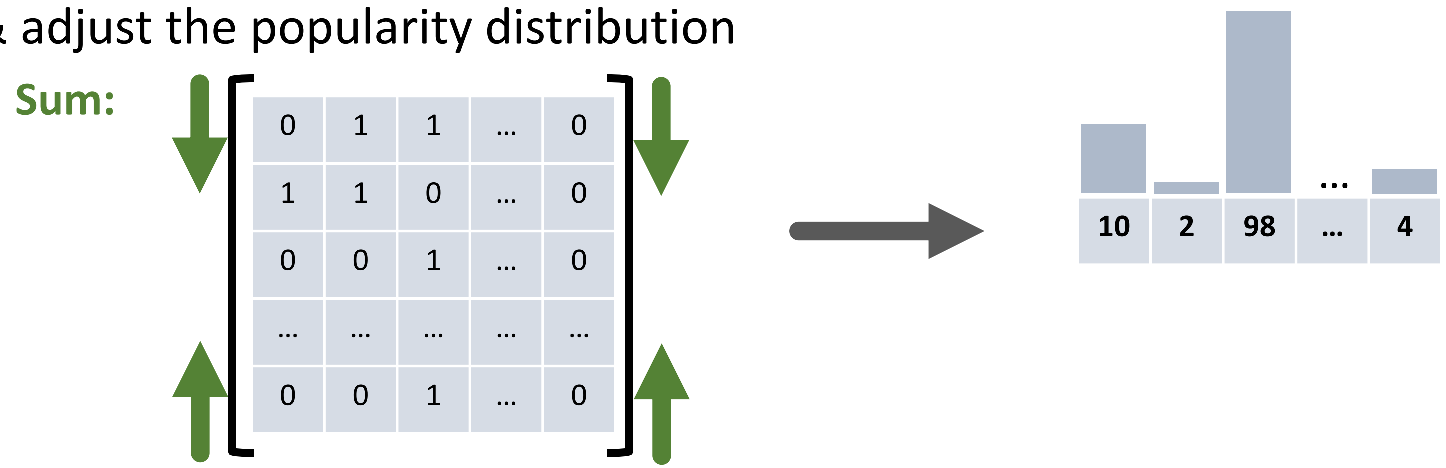
$$\begin{array}{r} \\ \textit{target user} \\ \\ \\ \\ \end{array} \begin{bmatrix} 0 & 1 & 1 & \dots & 0 \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \dots & \mathbf{0} \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 1 & \dots & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}$$

Exercise 1: POP Recommender

1. Save IDs of items seen by the target ***user***

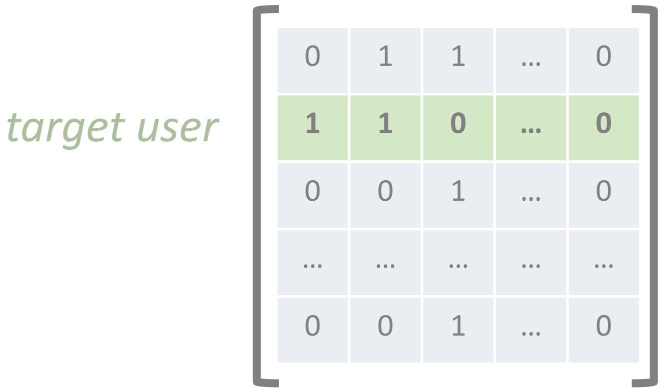


2. **Get** & adjust the popularity distribution

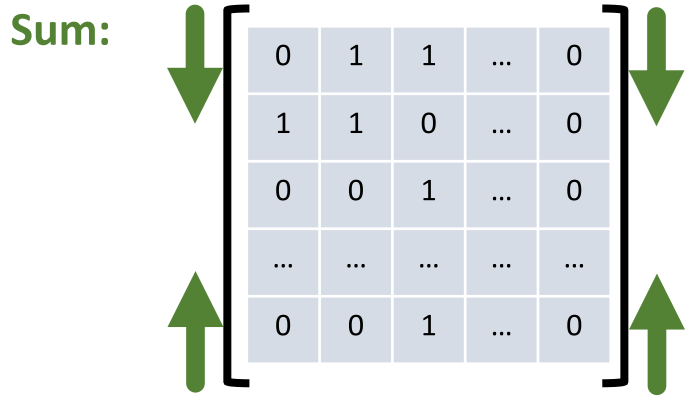


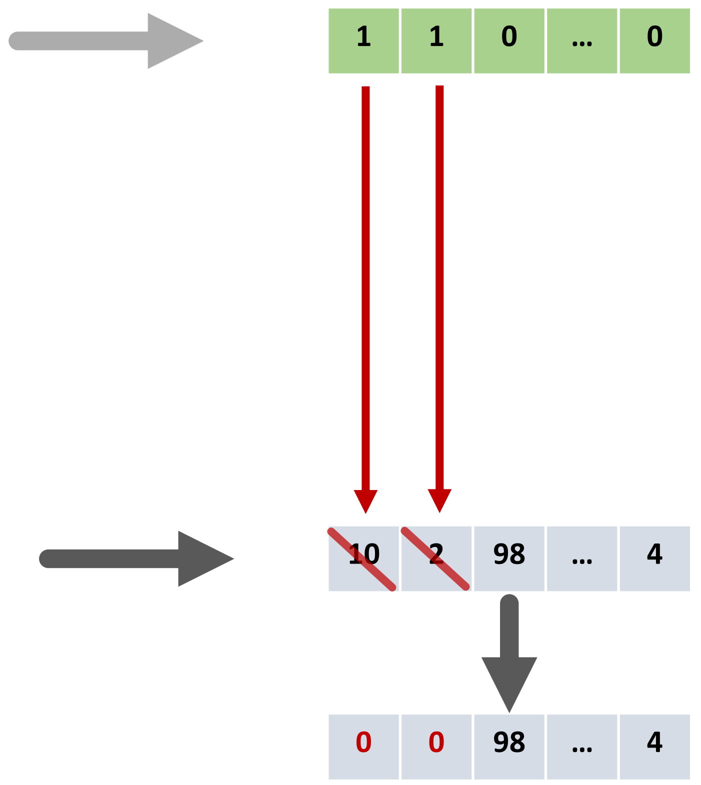
Exercise 1: POP Recommender

1. Save IDs of items seen by the target ***user***



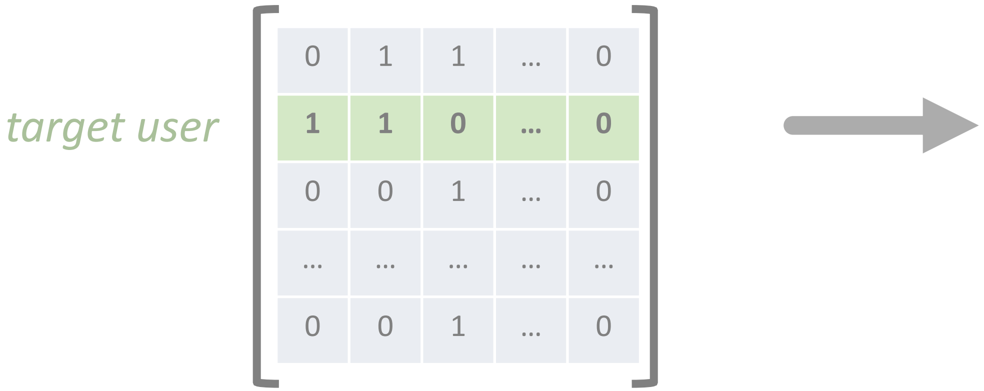
2. Get & **adjust** the popularity distribution

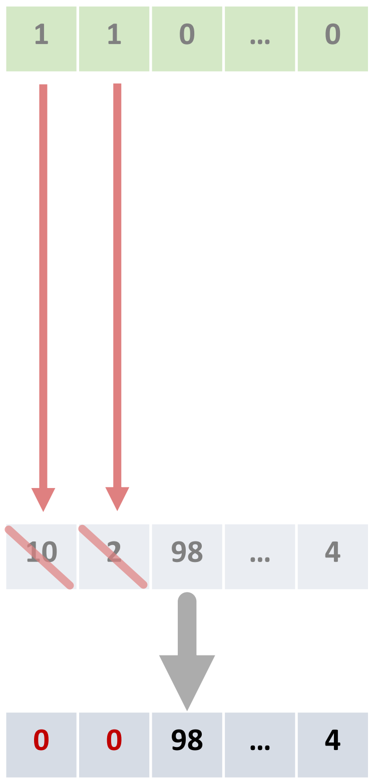




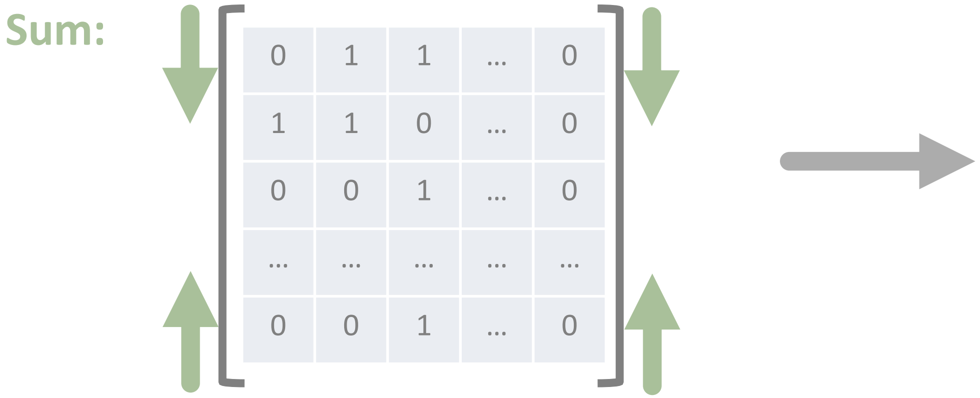
Exercise 1: POP Recommender

1. Save IDs of items seen by the target ***user***





2. Get & **adjust** the popularity distribution



3. Select top K most popular Items

Questions so far?

Exercise 2: Implicit Feedback - Reminder

Explicit feedback:
(e.g. ratings)

	Item_1	Item_2	Item_3	…
User_1	4.0	-	-	…
User_2	-	1.0	3.0	…
User_3	5.0	-	3.0	…
…	…	…	…	…

Today's lecture

Implicit feedback:
(inferred from interactions)

	Item_1	Item_2	Item_3	…
User_1	1	0	0	…
User_2	0	1	1	…
User_3	1	0	1	…
…	…	…	…	…

Used in the Exercise

Exercise 2: Item-Based CF

- End goal -- create a recommender system, that is able to suggest a list of K not seen before items to a target user.

Exercise 2: Item-Based CF

- End goal -- create a recommender system, that is able to suggest a list of K not seen before items to a target user.
- This time we use Item-Based approach to estimate how well an item fits to the taste of a user.

Exercise 2: Item-Based CF

- End goal -- create a recommender system, that is able to suggest a list of K not seen before items to a target user.
- This time we use Item-Based approach to estimate how well an item fits to the taste of a user.

1. We’ll learn to calculate the “fitness” **score** for each pair (**user**, **item**)
2. For the target user we create a **list** of all unseen items **ranked** by the “fitness” score and take **top K** of it

Exercise 2: CF Explicit vs Implicit

Item-Based Collaborative Filtering, Explicit feedback:

$$r'_{a,p} = \frac{\sum_{n \in N} sim(n,p) * r_{a,n}}{\sum_{n \in N} sim(n,p)}$$

Exercise 2: CF Explicit vs Implicit

Item-Based Collaborative Filtering, Explicit feedback:

$$r'_{a,p} = \frac{\sum_{n \in N} sim(n,p) * r_{a,n}}{\sum_{n \in N} sim(n,p)}$$

Item-Based *Collaborative Filtering*, Implicit feedback:

$$score(u,p) = \frac{1}{|N_p|} \sum_{n \in N_i} sim(n,p) * I(u,n)$$

$|N_p| \leq N$ N – number of neighbors to consider (hyper-parameter)

Exercise 2: CF Explicit vs Implicit

Item-Based Collaborative Filtering, Explicit feedback:

We want to calculate a score, abstract measure of fitness, not a rating.

$$r'_{a,p} = \frac{\sum_{n \in N} sim(n,p) * r_{a,n}}{\sum_{n \in N} sim(n,p)}$$

Item-Based *Collaborative Filtering*, Implicit feedback:

$$score(u,p) = \frac{1}{|N_p|} \sum_{n \in N_i} sim(n,p) * I(u,n)$$

$|N_p| \leq N$ N – number of neighbors to consider (hyper-parameter)

Exercise 2: CF Explicit vs Implicit

Item-Based Collaborative Filtering, Explicit feedback:

$$r'_{a,p} = \frac{\sum_{n \in N} sim(n,p) * r_{a,n}}{\sum_{n \in N} sim(n,p)}$$

In our scenario this is either 0 or 1

Item-Based *Collaborative Filtering*, Implicit feedback:

$$score(u,p) = \frac{1}{|N_p|} \sum_{n \in N_p} sim(n,p) * I(u,n)$$

Interaction indicator function

$|N_p| \leq N$ N – number of neighbors to consider (hyper-parameter)

Exercise 2: CF Explicit vs Implicit

Item-Based Collaborative Filtering, Explicit feedback:

$$r'_{a,p} = \frac{\sum_{n \in N} sim(n,p) * r_{a,n}}{\sum_{n \in N} sim(n,p)}$$

While predicting a score normalization is not necessary but we'll take a mean to account for the size of our data sample

Item-Based *Collaborative Filtering*, Implicit feedback:

$$score(u,p) = \frac{1}{|N_p|} \sum_{n \in N_p} sim(n,p) * I(u,n)$$

Number of actual neighbors can be smaller than N in our limited dataset

$|N_p| \leq N$ N – number of neighbors to consider (hyper-parameter)

Exercise 2: CF + Implicit: Algorithm

Recommendation for user ***u***:

- For every item ***p*** in the collection **not seen** by ***u*** calculate **score(u, p)**
- Rank the list according to the score, return IDs of the top K items

Exercise 2: CF + Implicit: Algorithm

Recommendation for user ***u***:

- For every item ***p*** in the collection **not seen** by ***u*** calculate **score(u, p)**
- Rank the list according to the score, return IDs of the top K items

Calculating **score(u, p)**:

- For every item ***h*** seen by ***u*** calculate **sim(h, p)**
- Rank the list and take top N neighbors, return the average

Exercise 2: CF + Implicit: Algorithm

Recommendation for user ***u***:

- For every item ***p*** in the collection **not seen** by ***u*** calculate **score(u, p)**
- Rank the list according to the score, return IDs of the top K items

Calculating **score(u, p)**:

- For every item ***h*** **seen** by ***u*** calculate **sim(h, p)**
- Rank the list and take top N neighbors, return the average

Calculating **sim(h, p)**:

- Jaccard Index
- The target user does not contribute to the similarity

Questions so far?

Task 1

```python
def getItemSim(inter: np.array, target_vec: np.array) -> np.array:
    """
    inter: np.array - matrix of target items.
    target_vec: int - vector to calculate similarity to.
        use jaccard index.

    returns: np.array - n similarity scores
    """

    closest_items = None
    item_similarities = None

    # TODO: YOUR IMPLEMENTATION

    return item_similarities
```

Task 2

```python
def getUserItemScore(inter: np.array,
                     target_user: int,
                     target_item: int,
                     n: int = 2) -> np.array:
    """
    inter: np.array - interaction matrix
    target_user: int - user to get Score for
    target_item: int - item to get Score for
    n: int - n most similar items contributing to the score

    returns: float - mean of similarity scores
    """

    item_similarities_mean = None

    # TODO: YOUR IMPLEMENTATION.

    return item_similarities_mean
```

Task 3

```python
def recTopK(inter_matr: np.array,
            user: int,
            top_k: int,
            n: int) -> (np.array, np.array):
    '''
    inter_matr - np.array from the task 1
    user - user_id, integer
    top_k - expected length of the resulting list
    n - n most similar items contributing to the score

    returns - list/array of top K popular items that the user has never seen
              (sorted in the order of descending popularity)
    '''

    top_rec = None
    scores = None

    # TODO: YOUR IMPLEMENTATION.


    return np.array(top_rec), np.array(scores)
```

Thank you & Good luck!