

# Exercises for Social Gaming and Social Computing (IN2241 + IN0040) – Introduction to **Exercise 3**



# Exercise Content

Sheet Number	Exercise	Working Time
1	<ul style="list-style-type: none"><li>• Introduction to Python: basic Python programming language exercises</li><li>• Graph Drawing using igraph</li></ul>	Monday, May 27 - Monday, June 3, 24:00
2	<ul style="list-style-type: none"><li>• Centrality measures</li></ul>	Monday, June 3 - Monday, June 17, 24:00
3	<ul style="list-style-type: none"><li>• Recommender Systems as an example for systems using simple forms of social context: Collaborative Filtering</li></ul>	Monday, June 17 - Monday, June 24, 24:00
4	<ul style="list-style-type: none"><li>• Inferring social tie strength from activity data in social networking platforms with linear regression</li></ul>	Monday, June 24- Monday, July 01, 24:00
5	<ul style="list-style-type: none"><li>• finding social groups with clustering methods on profile data (K-Means) and graph clustering (Girvan Newman method)</li></ul>	Monday, July 01 - Monday, July 08, 24:00
6	<ul style="list-style-type: none"><li>• analyzing short-term social context using mobile interaction data (Reality Mining)</li></ul>	Monday, July 08 - Monday, July 15, 24:00

## Task: Collaborative Filtering

- Task: **Complement the given code** for a simple **collaborative filtering** movie recommender algorithm (a standard social computing application).
- Dataset: **Movielens**: Predict the top twenty (yet unrated) movies for user 15!

The screenshot shows a text editor window titled 'u.item'. The file path is '~/Desktop/socialCompUGaming2...ceV2/exercise2/movielens/u.item'. The content of the file is a list of movie titles and their corresponding IMDb URLs, separated by a pipe character '|'. The list includes:

- 1|Toy Story (1995) |01-Jan-1995| |http://us.imdb.com/M/title-exact?Toy%20Story
- 2|GoldenEye (1995) |01-Jan-1995| |http://us.imdb.com/M/title-exact?GoldenEye
- 3|Four Rooms (1995) |01-Jan-1995| |http://us.imdb.com/M/title-exact?FourRooms
- 4|Get Shorty (1995) |01-Jan-1995| |http://us.imdb.com/M/title-exact?GetShorty
- 5|Copycat (1995) |01-Jan-1995| |http://us.imdb.com/M/title-exact?Copycat
- 6|Shanghai Triad (Yao a yao yao dao waipo qiao) (1995) |01-Jan-1995| |http://us.imdb.com/M/title-exact?ShanghaiTriad
- 7|Twelve Monkeys (1995) |01-Jan-1995| |http://us.imdb.com/M/title-exact?TwelveMonkeys
- 8|Babe (1995) |01-Jan-1995| |http://us.imdb.com/M/title-exact?Babe
- 9|Dead Man Walking (1995) |01-Jan-1995| |http://us.imdb.com/M/title-exact?DeadManWalking
- 10|Richard III (1995) |22-Jan-1996| |http://us.imdb.com/M/title-exact?RichardIII
- 11|Seven (Se7en) (1995) |01-Jan-1995| |http://us.imdb.com/M/title-exact?Seven

u.data

File Path ▾ : ~/Desktop/socialCompUGaming2...ceV2/exercise2/movielens/u.data

u.data

1|196 242 3 881250949

2186 302 3 891717742

322 377 1 878887116

4244 51 2 880606923

5166 346 1 886397596

6298 474 4 884182806

7115 265 2 881171488

8253 465 5 891628467

9305 451 3 886324817

106 86 3 883603013

1162 257 2 879372434

12286 1014 5 879781125

13200 222 5 876042340

14210 40 3 891035994

15224 29 3 888104457

16303 785 3 879485318

17122 387 5 879270459

18194 274 2 879539794

19291 1042 4 874834944

20234 1184 2 892079237

21119 392 4 886176814

22167 486 4 892738452

23299 144 4 877881320

24291 118 2 874833878

25308 1 4 887736532

2695 546 2 879196566

2738 95 5 892430094

28102 768 2 883748450

2963 277 4 875747401

30160 234 5 876861185

3150 246 3 877052329

32301 98 4 882075827

33225 193 4 879539727

34290 88 4 880731963

# Task: Collaborative Filtering

$r =$

	items								
users	4	—	—	—	—	—	1	2	—
	—	—	—	5	—	—	1	—	—
	—	5	5	—	—	—	1	—	5
	—	9	—	—	9	—	—	—	—
	—	—	—	—	—	3	3	—	—
	—	8	5	—	6	—	—	—	5
	1	—	—	1	—	3	—	2	—
	—	—	—	—	—	—	0	—	—
	—	7	6	1	6	—	—	—	6
	—	—	—	1	—	—	9	4	—
	—	—	—	—	—	4	—	—	6

# Task: Collaborative Filtering

items

$r =$

users

4	—	—	—	—	—	1	2	—
—	—	—	5	—	—	1	—	—
—	5	5	—	?	—	1	—	5
—	9	—	—	9	—	—	—	—
—	—	—	—	—	3	3	—	—
—	8	5	—	6	—	—	—	5
1	—	—	1	—	3	—	2	—
—	—	—	—	—	—	0	—	—
—	7	6	1	6	—	—	—	6
—	—	—	1	—	—	9	4	—
—	—	—	—	—	4	—	—	6

user u

item i

# Task: Collaborative Filtering

items

users

$r =$

4	—	—	—	—	—	1	2	—
—	—	—	5	—	—	1	—	—
—	5	5	—	?	—	1	—	5
—	9	—	—	9	—	—	—	—
—	—	—	—	—	3	3	—	—
—	8	5	—	6	—	—	—	5
1	—	—	1	—	3	—	2	—
—	—	—	—	—	—	0	—	—
—	7	6	1	6	—	—	—	6
—	—	—	1	—	—	9	4	—
—	—	—	—	—	4	—	—	6

user u

item i

$\mathcal{N}_i(u)$ : users that rated item i  
and that are similar to user u, e.g.:

$$\mathcal{N}_i(u) = \{v^{(i)} | \text{sim}(u, v^{(i)}) > \alpha\}$$

where e.g.

$$\text{sim}(u, v^{(i)}) = w_{uv} = \cos(\mathbf{u}, \mathbf{v}^{(i)}) \sim \mathbf{u} * \mathbf{v}^{(i)}$$

or:

$$\text{sim}(u, v^{(i)}) = w_{uv} = \frac{1}{1 + |\mathbf{u} - \mathbf{v}^{(i)}|^2}$$

# Task: Collaborative Filtering

items

users

$r =$

4	—	—	—	—	—	1	2	—
—	—	—	5	—	—	1	—	—
—	5	5	—	?	—	1	—	5
—	9	—	—	9	—	—	—	—
—	—	—	—	—	3	3	—	—
—	8	5	—	6	—	—	—	5
1	—	—	1	—	3	—	2	—
—	—	—	—	—	—	0	—	—
—	7	6	1	6	—	—	—	6
—	—	—	1	—	—	9	4	—
—	—	—	—	—	4	—	—	6

user u

$\mathcal{N}_i(u)$

$\mathcal{N}_i(u)$ : users that rated item i and that are similar to user u, e.g.:

item i

$$\mathcal{N}_i(u) = \{v^{(i)} | sim(u, v^{(i)}) > \alpha\}$$

where e.g.

$$sim(u, v^{(i)}) = w_{uv} = \cos(\mathbf{u}, \mathbf{v}^{(i)}) \sim \mathbf{u} * \mathbf{v}^{(i)}$$

or:

$$sim(u, v^{(i)}) = w_{uv} = \frac{1}{1 + |\mathbf{u} - \mathbf{v}^{(i)}|^2}$$

## Task: Collaborative Filtering

	items								
	4	-	-	-	-	-	1	2	-
	-	-	-	5	-	-	1	-	-
user u	-	5	5	-	?	-	1	-	5
	-	9	-	-	9	-	-	-	-

now: **predicted rating** for  
item  $i$  of user  $u$

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv} (r_{vi} - \bar{r}_v)}{\sum_{v \in \mathcal{N}_i(u)} w_{uv}}$$

or (more simple):

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv} r_{vi}}{\sum_{v \in \mathcal{N}_i(u)} w_{uv}}$$

$\mathcal{N}_i(u)$ : users that rated item  $i$   
and that are similar to user  $u$ , e.g.:

$$\mathcal{N}_i(u) = \{v^{(i)} | sim(u, v^{(i)}) > \alpha\}$$

where e.g.

$$\text{sim}(u, v^{(i)}) = w_{uv} = \cos(\mathbf{u}, \mathbf{v}^{(i)}) \sim \mathbf{u} * \mathbf{v}^{(i)}$$

or:

$$\text{sim}(u, v^{(i)}) = \frac{w_{uv}}{\mathbf{1} + \|\mathbf{u} - \mathbf{v}^{(i)}\|^2}$$



# Problem 3.1 Implementing a Collaborative Filtering Based Movie Recommender System

## Problem 3.1: Movie Recommender System

You will work with subset of the [MovieLens](#) dataset which contains 1862 movies and 100k ratings. The input is provided in two data files, a listing of movies (*u.item*) and their ratings by users (*u.data*).

For the *u.item* file, each row represents a movie with its attributes separated by '|' but the only attributes that matter for the exercise are the movie ID and the movie name. For the *u.data* file, each row represents one user's rating for a movie. The attributes are:

- user ID
- movie ID
- rating
- timestamp (not relevant for the exercise)

Your task will be to implement a collaborative filtering recommender for the given dataset.

## Collaborative Filtering Recommender

The main goal is to find a list of users similar to the target user for recommendations. There are several ways of measuring similarity, e.g. using the cosine similarity or the Euclidean distance. For simplicity reasons, the latter will be used. The estimated rating for the target user for each movie is given by

$$r_{u,i} = \frac{\sum_{v \in N_i(u)} w_{u,v} r_{v,i}}{\sum_{v \in N_i(u)} w_{u,v}}$$

where  $r_{u,i}$  is the estimated recommendation of item  $i$  for target user  $u$ .  $N_i(u)$  is the set of similar users to target user  $u$  for the designated item  $i$ .  $w_{u,v}$  is the similarity score between users  $u$  and  $v$  (used as a weighting factor).

## Task Overview

**Write a Python program that implements a simple collaborative filtering movie recommender.** The entry point to your recommendation engine should be a function that takes a user ID, and paths to the two aforementioned datasets. The function should return the top twenty recommended movies for that user (which were not rated by the user yet). The output should be a list of tuples, containing a movie and its estimated rating - sorted by the highest rating. You can either use the templates given below or design your recommendation engine the way you want, but straightforward collaborative filtering is highly recommended. Make sure that the code is clean, readable, and well documented. **Test your program for user 15.**

# Problem 3.1 Implementing a Collaborative Filtering Based Movie Recommender System

---

## Task 1: Creating The Dictionaries

The first thing to do is **implementing two functions that create the dictionaries** for the movies and the user ratings. For the movies, use the movie ID as a key and the movie name as the value. Concerning the user ratings dictionary, a dict of dicts format like `{u1_id: {m1_id: rating1, m2_id: rating2, ...}, u2_id: {...}, ...}` might be useful since the user's rating needs to be stored for each movie.

---

## Task 2: Calculating Similarity Scores

The task is to **implement a function that calculates the similarity score**  $w_{u,v}$  between two users  $u$  and  $v$ . This should be done for all users and the movies they both rated. For the similarity score, use the *normalized* Euclidean distance measure given by

$$w_{u,v} = \frac{1}{1 + \sum_i (r_{u,i} - r_{v,i})^2}$$

---

## Task 3: The Recommender System

Now it is time to **implement the collaborative filtering recommender system that outputs the recommended movies for a target user**. Use the functionality defined above to compute the similarity scores between them and calculate the resulting estimated ratings. Sort the movies according to the highest rating before returning them.

---

# Submitting Your Solution

---

- work by **expanding** the .ipynb iPython notebook for the exercise that you downloaded from Moodle.
- save your expanded .ipynb iPython notebook in your working directory. Submit your .ipynb iPython notebook **via Moodle** (nothing else)
- remember: working in groups is not permitted. Each student must submit **her own** ipynb notebook!
- we check for **plagiarism**. Each detected case will have the consequence of 5.0 for the whole exercise grade.
- **deadline**: please check slide nr 2 (this slide-set)



# Citations

---

Desrosiers, C., & Karypis, G.: A Comprehensive Survey of Neighborhood-based Recommendation Methods, 2011 in: Ricci et al (eds.) "Recommender Systems Handbook", Springer 2011