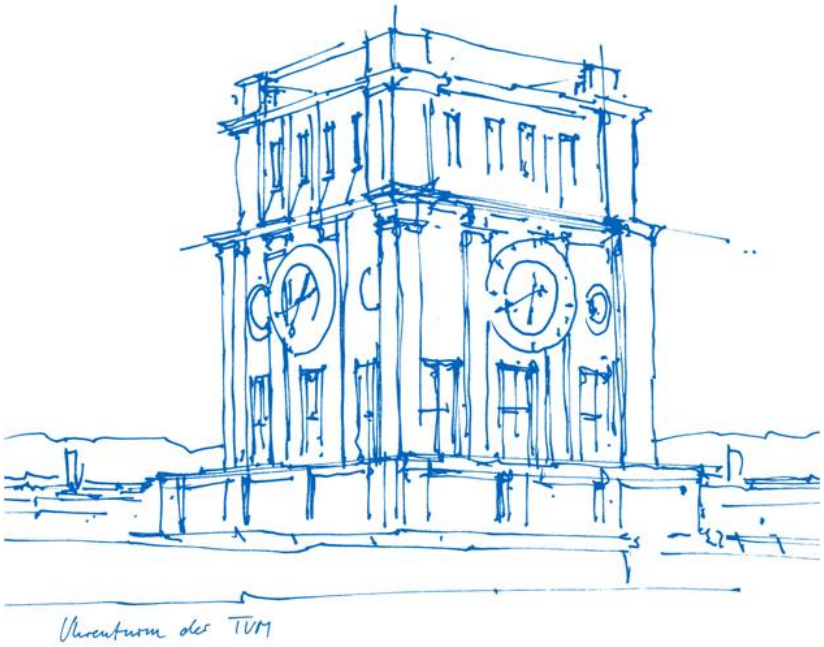
# **Exercises** for Social Gaming and Social Computing (IN2241 + IN0040) – Introduction to Exercise 3



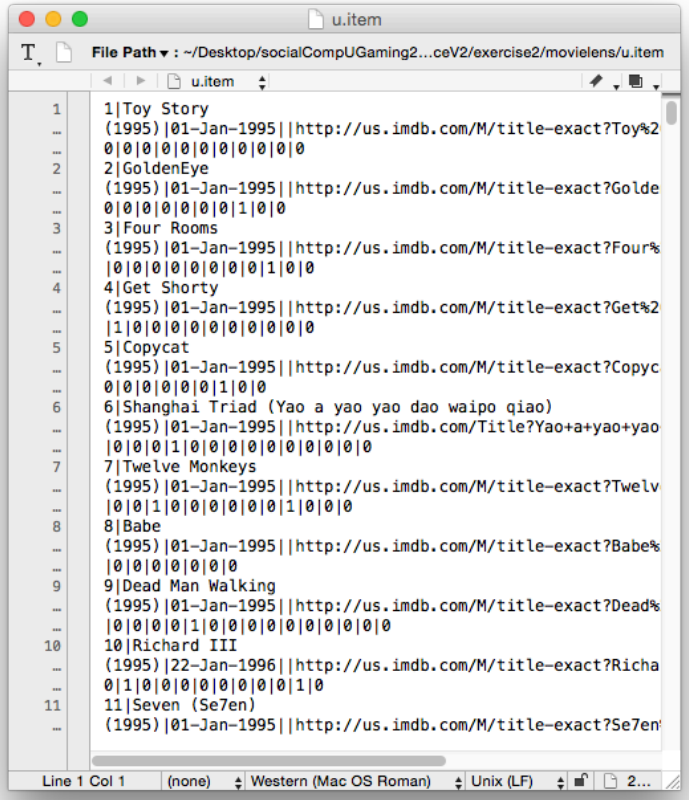
# Exercise Content

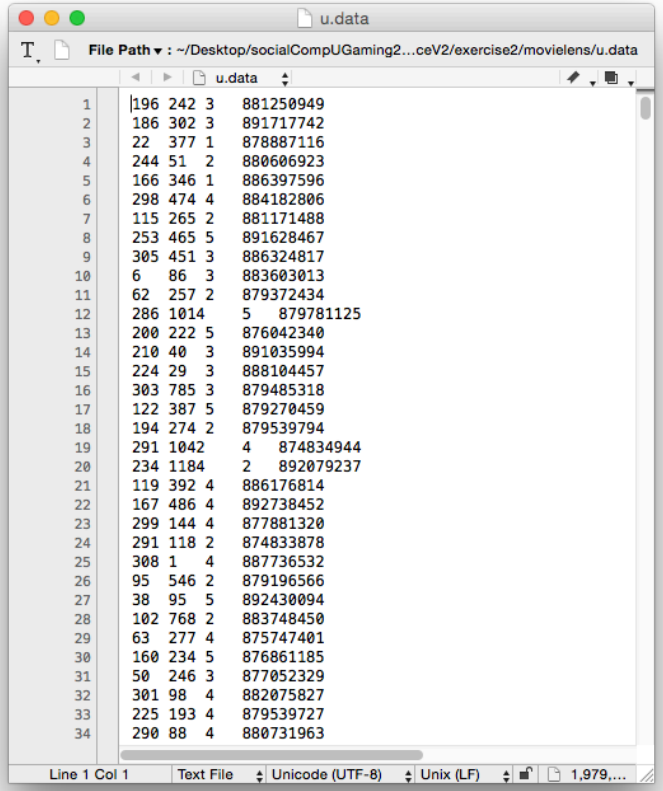
| Sheet Number | Exercise | Working Time |
|---|---|---|
| 1 | • Introduction to Python: basic Python programming language exercises<br>• Graph Drawing using igraph | Monday, May 27 - Monday, June 3, 24:00 |
| 2 | • Centrality measures | Monday, June 3 - Monday, June 17, 24:00 |
| 3 | • Recommender Systems as an example for systems using simple forms of social context: Collaborative Filtering | Monday, June 17 - Monday, June 24, 24:00 |
| 4 | • Inferring social tie strength from activity data in social networking platforms with linear regression | Monday, June 24- Monday, July 01, 24:00 |
| 5 | • finding social groups with clustering methods on profile date (K-Means) and graph clustering (Girvan Newman method) | Monday, July 01 - Monday, July 08, 24:00 |
| 6 | • analyzing short-term social context using mobile interaction data (Reality Mining) | Monday, July 08 - Monday, July 15, 24:00 |

# Task: Collaborative Filtering

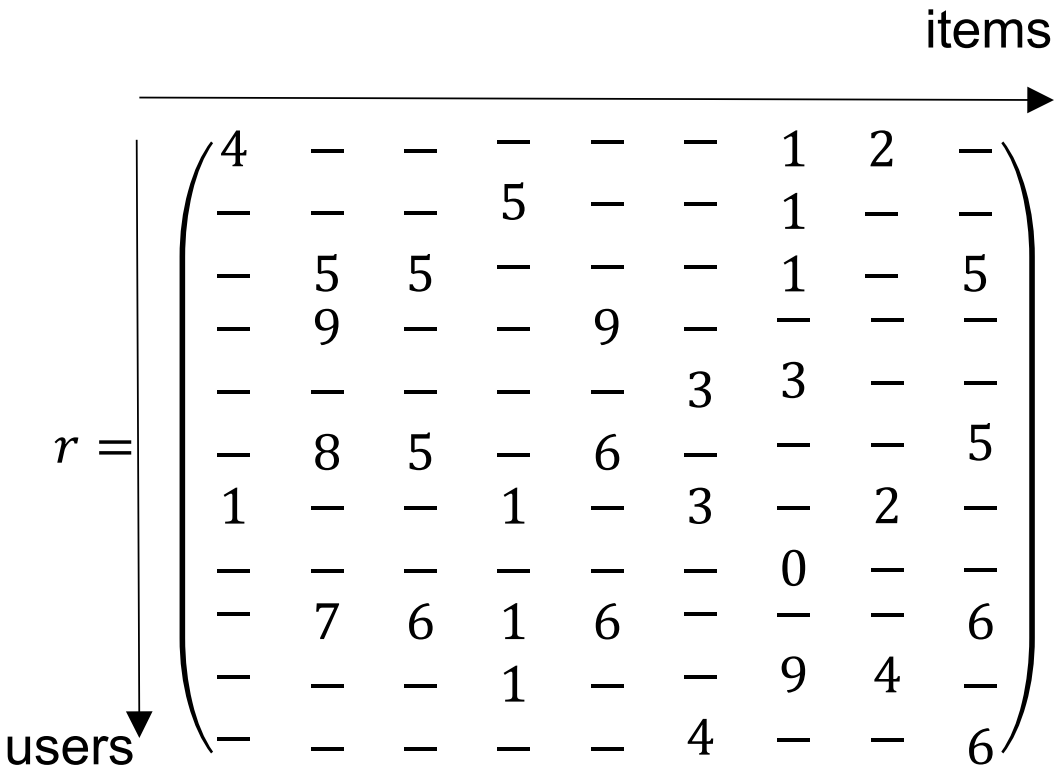
- Task: Complement the given code for a simple collaborative filtering movie recommender algorithm (a standard social computing application).

- Dataset: Movielens: Predict the top twenty (yet unrated) movies for user 15!

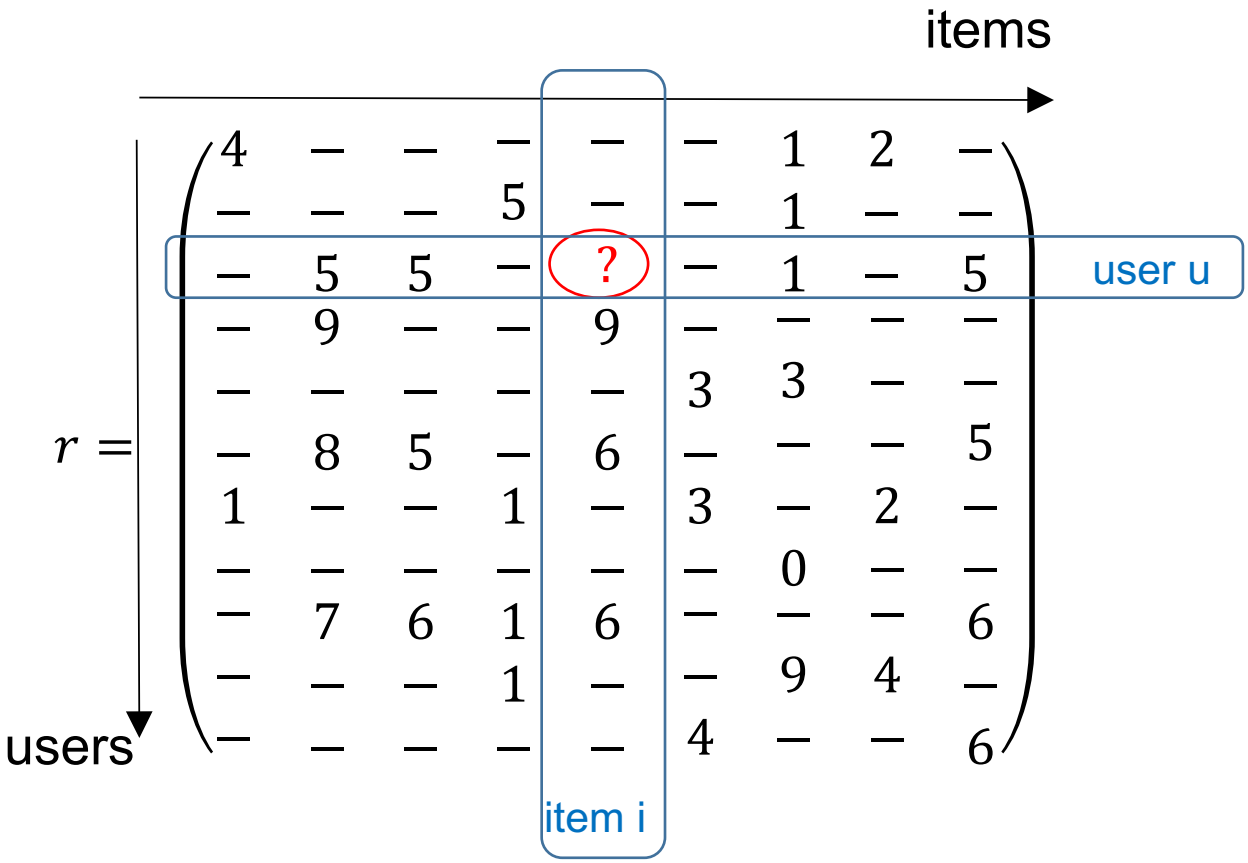


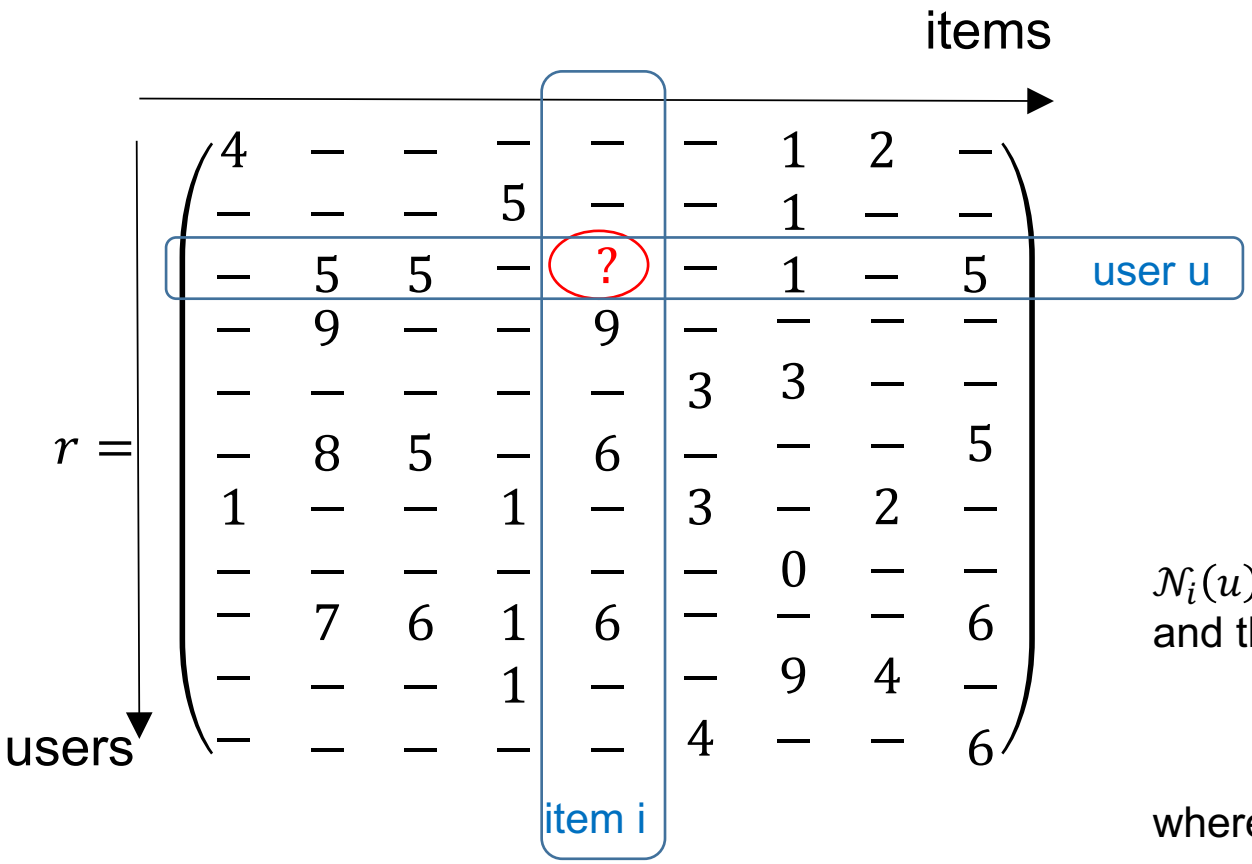
# Task: Collaborative Filtering



# Task: Collaborative Filtering



# Task: Collaborative Filtering




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

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

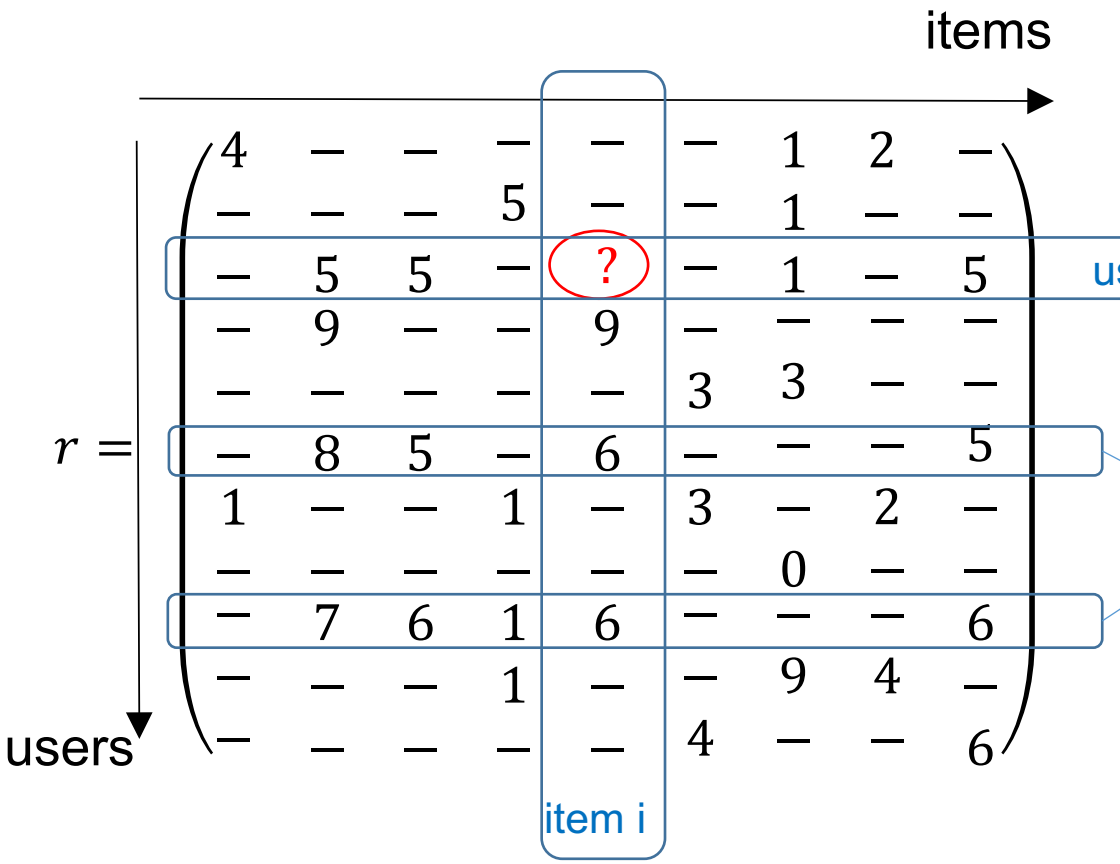
where e.g.

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

or:

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

# Task: Collaborative Filtering




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

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

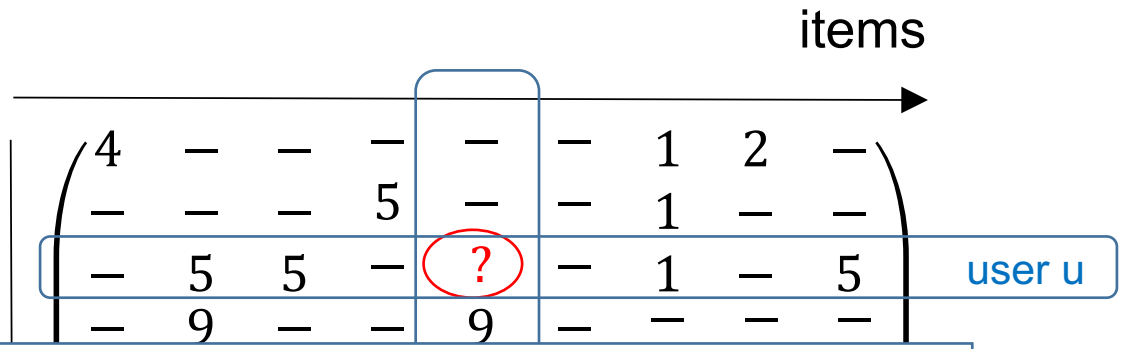
where e.g.

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

or:

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

# Task: Collaborative Filtering




now: predicted rating for item i of user u

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

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.:

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

where e.g.

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

or:

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

# 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 collborative 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: rating 2, ...}, 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 collabortive 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