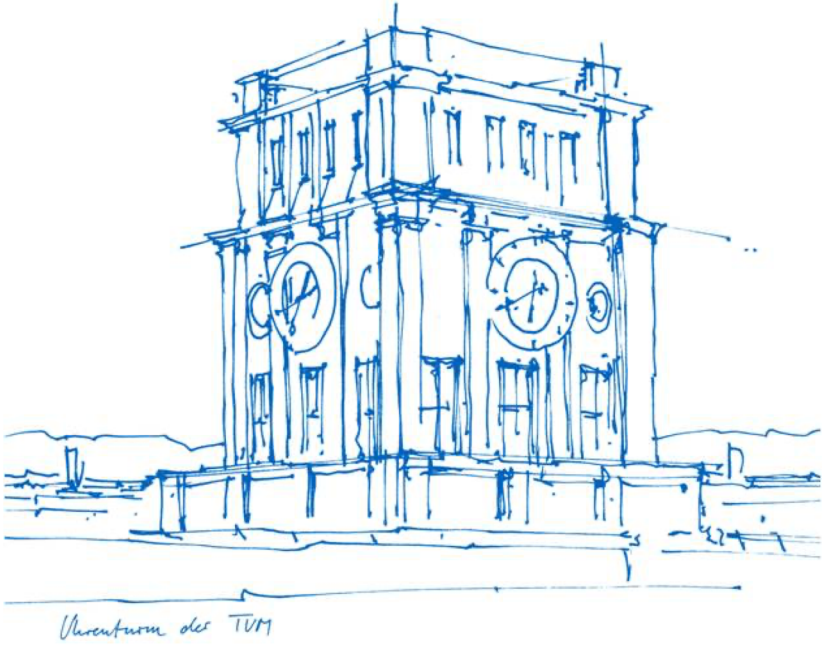
Exercises for
Social Gaming and
Social Computing
(IN2241 + IN0040) – Introduction to

Exercise Sheet 3
Collaborative Filtering with Steam Games



Exercise Content

Sheet Number	Exercise	Working Time
1	• Introduction to Python: basic Python programming language exercises • Graph Drawing using network, introduction to Steam API	Monday, May 27 - Monday, June 3, 24:00
2	• Centrality measures	Monday, June 3 - Monday, June 17, 24:00
3	• Collaborative Filtering Recommender System using Steam data	Monday, June 17 - Monday, June 24, 24:00
4	• Trying to infer toxic behavior from game data in DotA 2	Monday, June 24- Monday, July 01, 24:00
5	• Clustering EVE and BF2 players according to Radoff's motivational player types using K-means	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

Exercise Sheet 3: Collaborative Filtering with Steam Games

- goal: look at the games your friends play the most to figure out which ones you would love to play yourself
- build a collaborative filtering recommender system using the data we gather from your own Steam profile
- collaborative filtering is based on a rating to determine the similarity between users.
- playtime is usually the most authentic metric of enjoyment for gamers
    - therefore, we will use playtime as our rating system.

The Data: Your friends' Steam library

- use steam Web-API to fetch the games your friends and *their* friends play.
- if you prefer not to use your own Steam ID, you can use the default ID we provide
  - we recommend to use your own ID though

Task 3.1: Obtaining the data

1. Your first task is to gather the data needed to create the recommender system. Create a data structure that holds the needed information for each player and game.
Note: You cannot obtain a list from your profile with the Steam API unless your profile is set to public.

If you do not have a Steam profile, you can use the default values. However, we encourage you to use your own profile.

Hint: To obtain the games a user owns, use this: `games = data['response']['games']`. This returns a list of games, including the playtime (in minutes) which can be retrieved like this: `playtime = game['playtime_forever']` , where game refers to an item from the list of games.

Task 3.2: Association rule mining

The concept of association rule mining is rather simple: Looking at an itemset, one tries to find dependencies between items that could "belong together". A common example would be buying food at the store: If, for example, meat and salt are bought together often, but meat without salt not that often, it is assumed that there is a connection between those two. For games, if it was found that most of the users who own the demo version of a game also own the full version of that game, it would be a reasonable assumption that these users liked the demo and therefore bought the full version.

1. Your task here is to first convert the dictionary you created into a list of lists as this is the input required for the algorithm to work. Then, print out the most frequent items using the `min_support` attribute. Finally, print out the association rules and play around with the threshold value to get a reasonable amount of rules.

2. Discuss your results and try to answer the following questions: What kind of recommendations can be made? What does a confidence of 1.0 mean and is it meaningful for recommending games? Can you spot a correlation between the games with the highest support and the rules with the highest confidence? How does this affect the lift?
Hint: Play around with the threshold values until you get a reasonable amount (4-30) rows as output.

Association rules

- set of users $U = \{u_i\}$ and a set of games $G = \{g_j\}$.
- each user u_i likes a subset of games G_i :

User id	Game g_1	Game g_2	Game g_3	Game g_4	Game g_5	
1	yes		yes			$G_1 = \{g_1, g_3\}$
2		yes	yes		yes	$G_2 = \{g_2, g_3, g_5\}$
3		yes		yes		$G_3 = \{g_2, g_4\}$
4		yes			yes	$G_4 = \{g_2, g_5\}$

- Association Rule: if set of games X is liked by the users then a set of games Y is also liked ($X \rightarrow Y$). Example: $\{g_2, g_4\} \rightarrow \{g_1\}$ (Shopping basket analysis: users = shopping baskets; games = items)

Association rules

- The most important metrics for rules are support, confidence and lift.
- Support: the proportion of users that like game-set X

$$supp(X) = \frac{|\{u_i | X \subseteq G_i\}|}{|U|}$$

- Confidence of a rule $X \rightarrow Y$: proportion of users liking X that also like Y

$$conf(X \rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$$

- Lift of a rule $X \rightarrow Y$: mutual information of X and Y (how often do they appear together compared to how often that would be the case if they were independent .

$$lift(X \rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) * supp(X)}$$

In terms of probabilities of the events x: "X is liked" and y: "Y is liked" this would be $\frac{P(x,y)}{P(x) * P(y)} = \frac{P(x \cap y)}{P(x) * P(y)}$

Task 3.3: The Recommender System: Similarity Score

Finally, it is time to build the recommender system.

1. The first thing to do is to implement a similarity score that will be used to predict a user's playtime of an unowned game. We implement a similarity score between two users by taking the relative distance between two players. We use the following formula:

$$d(u, v) = \sum_{i \in \text{common\_games}} \frac{|r_{u,i} - r_{v,i}|}{r_{v,i}}$$

Where u and v are users and $r_{u,i}$ is the playtime of user u for game i.

You can then return the similarity with

$$w_{u,v} = \frac{1}{1 + d(u, v)}$$

Note: If no common games exist return 0.

Task 3.4: Recommender System: Predict ratings

With the similarity score calculated, we can now predict a user's playtime for games they don't own.

1. First, we create a set of all games, but we delete all games that are owned by less than 3 players. The reason is simple: If only 1 or 2 players own a game, it is impossible to derive a meaningful prediction since there is not enough data.

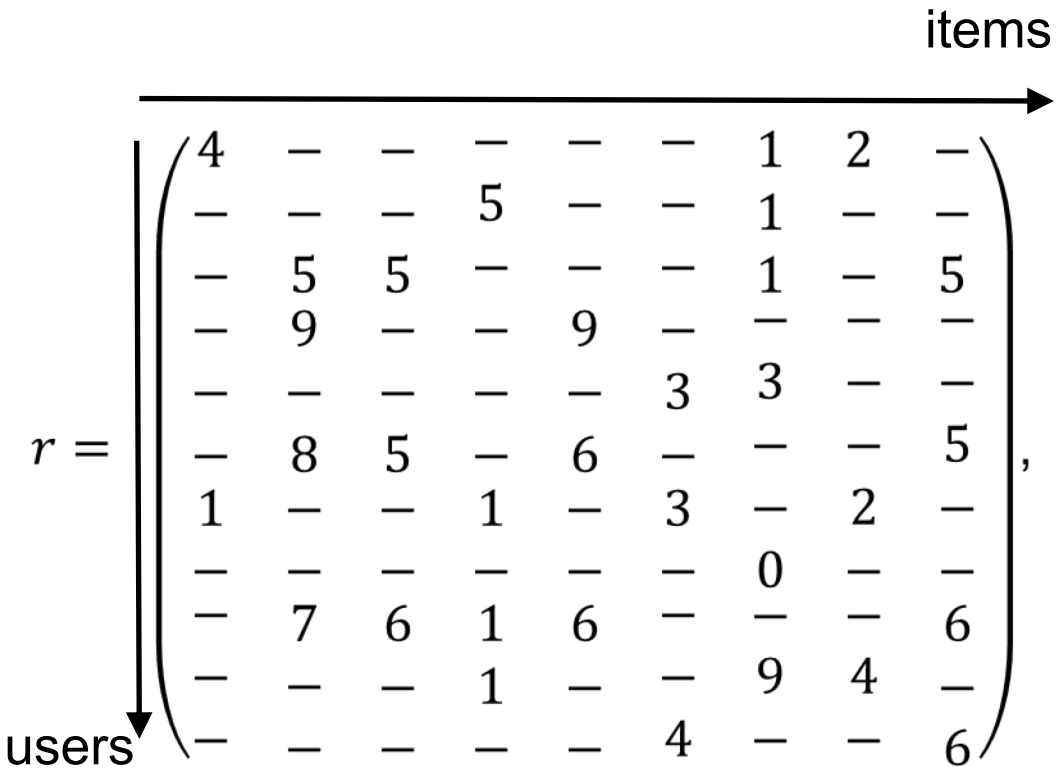
The predicted playtime for a game works analogous to the predicted rating of a movie/item in a conventional collaborative filtering recommender system:

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

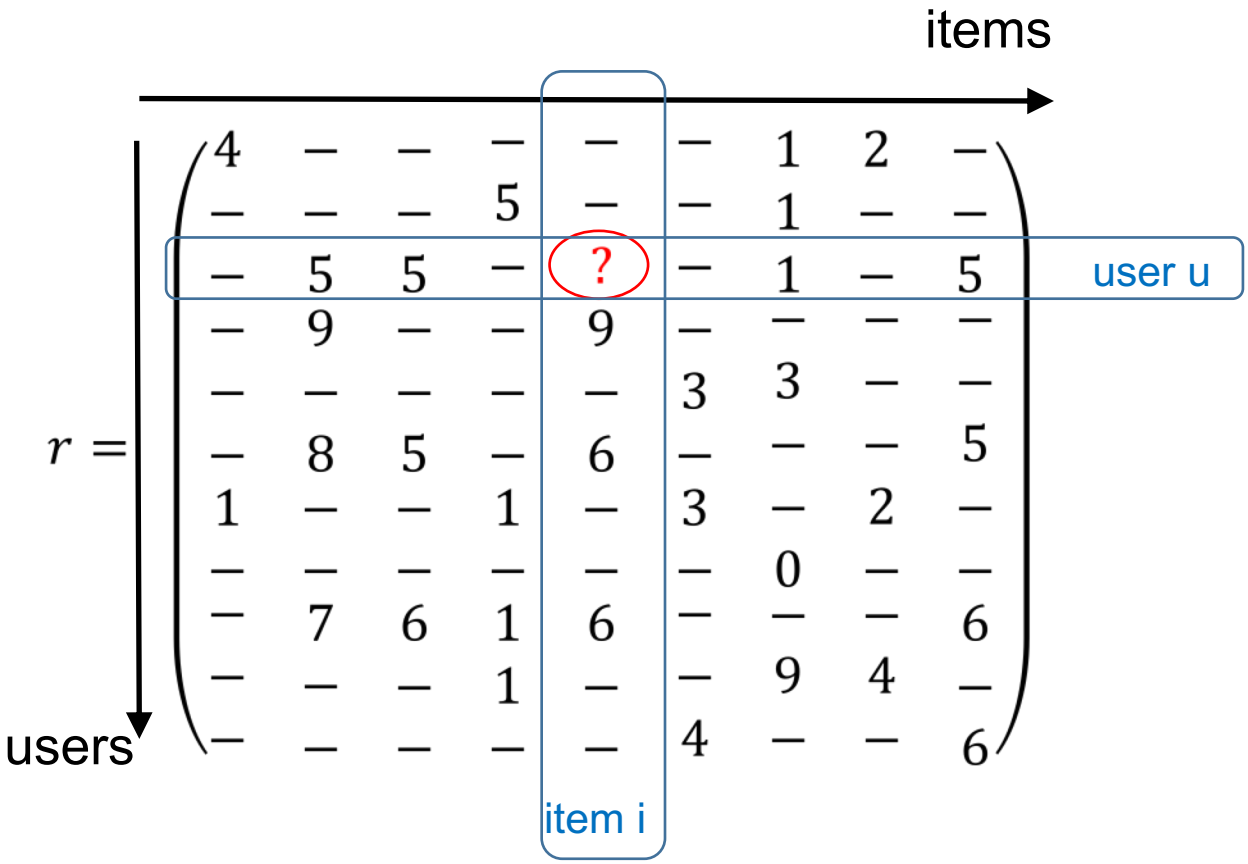
where

- $r_{u,i}$ 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).

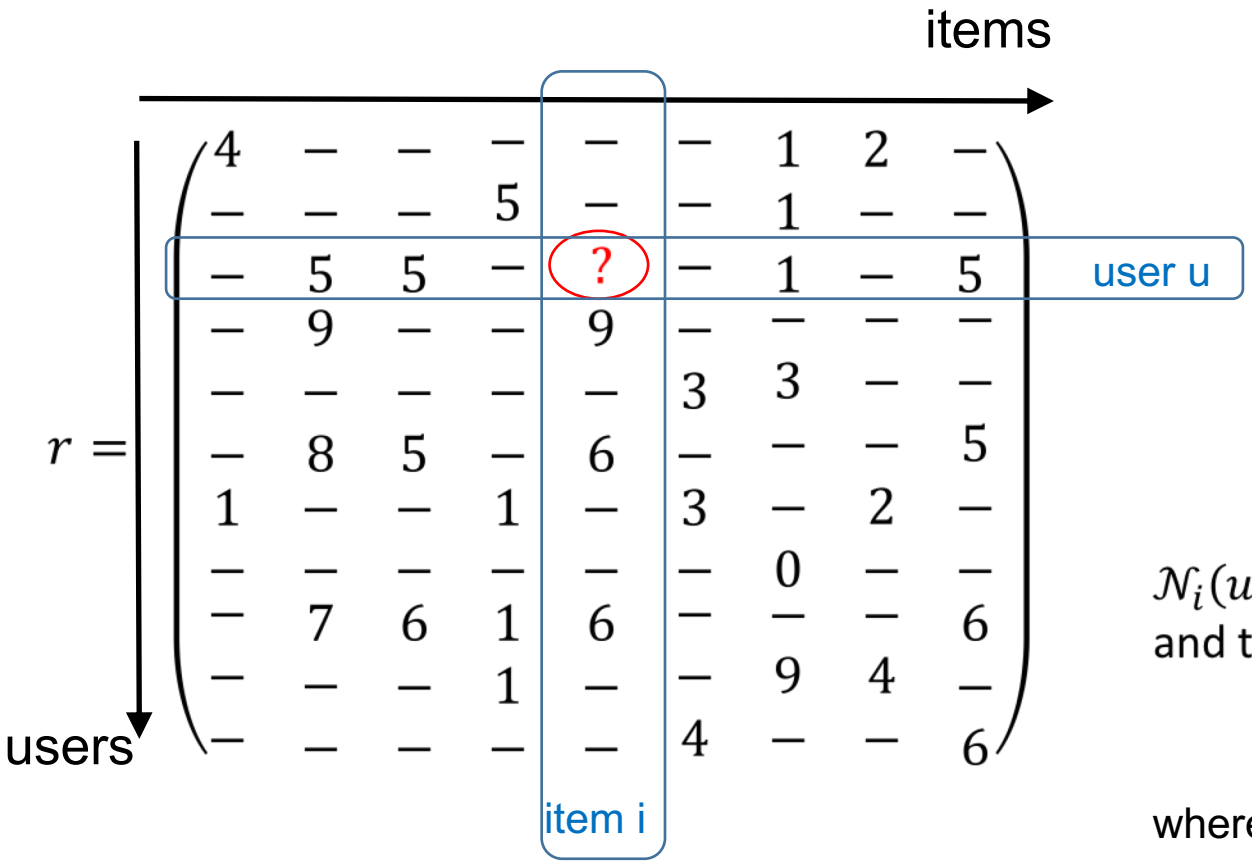
Collaborative Filtering



Collaborative Filtering (cont.)



Collaborative Filtering (cont.)



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

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

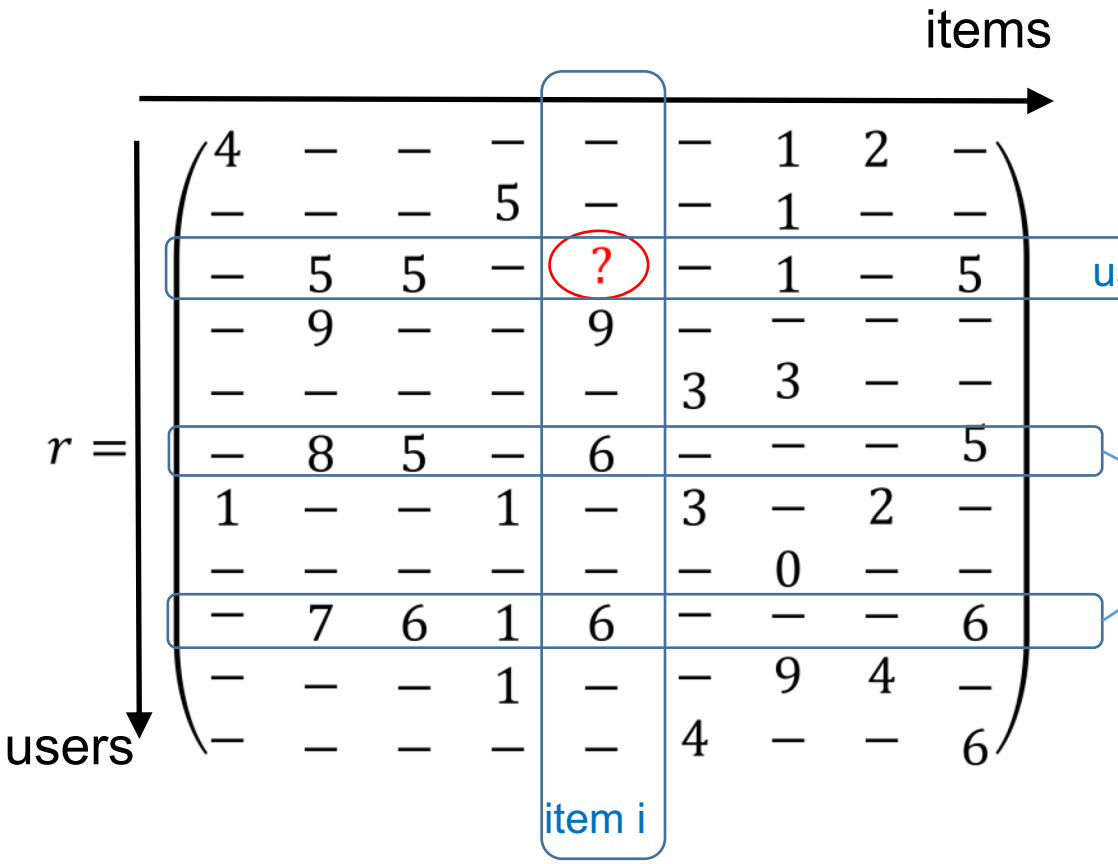
where e.g.

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

or:

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

Collaborative Filtering (cont.)



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

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

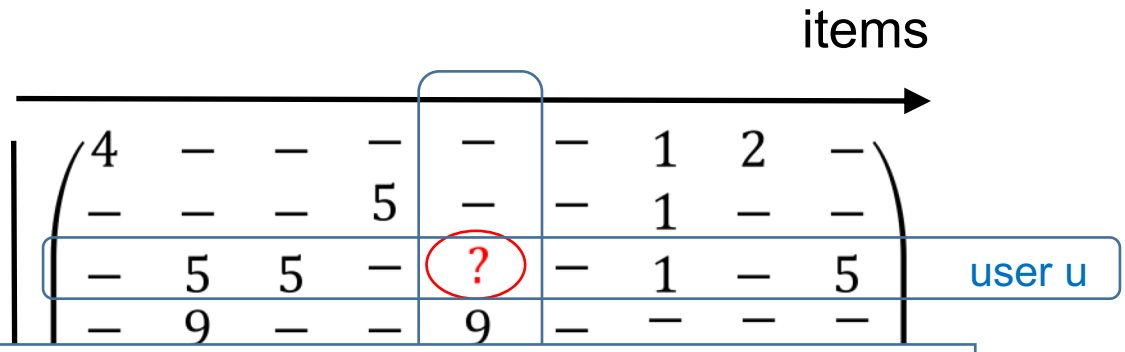
where e.g.

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

or:

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

Collaborative Filtering (cont.)



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:

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

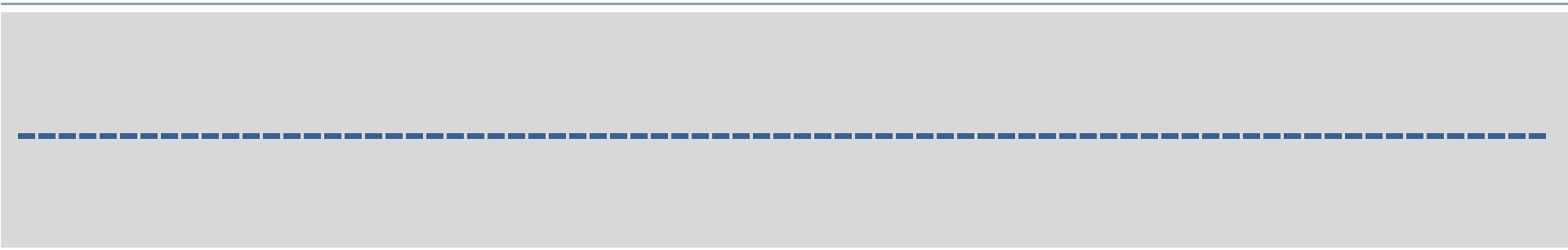
Task 3.5: Recommender System: Discussion

1. Sort the predicted ratings by estimated playtime (highest first) and print out the top 5 predictions for you (or the default user if you are using the default ID).

2. Discuss the difference in recommendations between the collaborative filtering approach and the association rule approach. Would you consider one more accurate than the other? Why/why not?

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 their own .ipynb notebook!
- we check for plagiarism. Each detected case will be graded with 5.0 for the whole exercise
- deadline: check Moodle



Literature

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