Board Game Recommendation System with Collaborative Filtering

Vanessza Tőke
Alba Regia Technical Faculty
Óbuda University
Székesfehérvár, Hungary
tvanessza@stud.uni-obuda.hu

Rozália Lakner
Alba Regia Technical Faculty
Óbuda University
Székesfehérvár, Hungary
lakner.rozalia@amk.uni-obuda.hu

***Abstract*—This paper presents the building of a recommendation system, based on collaborative filtering, using real data about board games and their ratings from users. It reviews the notion of web scraping, data cleaning and recommendation systems, and the used technology for development. Describes the building steps of a python-based application. Shows how to collect data using web scraping technique. Also mentions how to prepare, analyze, and visualize the gathered data with popular python libraries.**

Keywords—python, recommendation system, collaborative filtering, web scraping, data cleaning

I. Introduction

Nowadays, playing board games is popular than ever. The board games already existed a millennium ago, used in the ancient time, but their main purpose was more strategical, than fun. In the last few years, the number of game releases has increased, so from year-to-year it is harder to decide which games should we purchase, which ones are fitting best to our appetite. In Fig. 1. the number of released games can be seen in an annual breakdown, for which the data about boardgames collected from BoardGameGeeks.com [1].

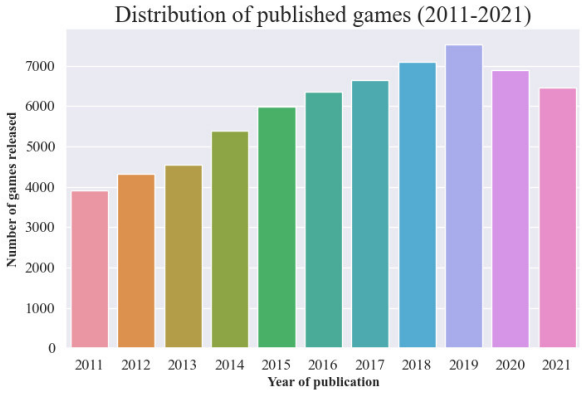

Fig. 1. Distribution of published board games in last 10 years

Thanks to the proliferation of high-performance computers and the developing information technology, the Big Data projects - working with huge amount of data, with the shortest possible processing time -, thus, the recommendation systems spread and used in a more widely scale.

For building recommendation systems, the most popular programming language is Python, and for making the development phase much easier, many python-based libraries are available to develop with more easily and faster. This article describes the process of building board game recommendation system, guiding through the basic concepts involved, using popular recommendation technique, the collaborative filtering and through the earlier steps like collecting and preparing the data with popular data science libraries.

II. Recommendation System

Recommendation systems have become a defining part of today's world. Many companies, like Netflix, Amazon, and Facebook use recommendation systems to get more customers to buy their products/services, or to made them to purchase something directed contents and advertisements.

According to the definition recommendation systems' main function is to suggest items for users according to their preferences, knowing their history for the recommendation [2]. These suggestions can be used in any kind of decision-making processes. The word *item* is a general term used to express what the system recommends to the users.

A. Types of recommending systems

For creating powerful and accurate systems we need as much data as possible, and preferably containing ratings from existing users and properties about the recommended item(s). That which kind of data will be more important from a recommendation point of view will be dependent on the type of used recommendation technique. The recommendation systems have different types, according to what preferences are mainly used for the recommendations. The mainly spread types are the collaborative filtering, the content-based filtering, and the hybrid processing.

1) Collaborative filtering

For collaborative filtering process the data about the users' rating habits are needed. If the data is enough for discover correlation between the similar users or the similar items, then the explored similarity can be used to recommend items to a given user according to what other users liked. The system's suggestion completely based on the ratings [3].

Within collaborative filtering the memory-based and model-based filtering are differentiated. The memory-based filtering has two types, which are be presented in this paper.

2) User-based collaborative filtering:

Fig. 2. User-based collaborative filtering method example, based on board game items. The recommendations based on the similar users' ratings.

The user-based collaborative filtering algorithm produces recommendation list for given user according to the view of other users, as it can be seen in Fig. 2. The basic concept is that if the ratings of rated items are similar between users, then it can be said that these users would rate other items similarly as well [4].

3) Item-based collaborative filtering:

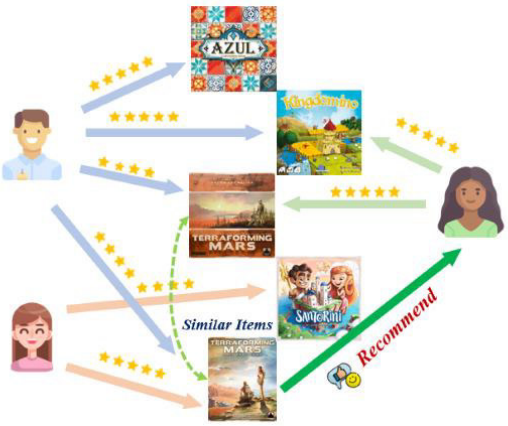

Fig. 3. Item-based collaborative filtering method example based on board game items. The recommendation based on similiraty between rated games.

The item-based collaborative filtering algorithm based on similarity, just like user-based filtering, but while there the similarity between the users is searched, using ratings, here among the rated items the similarity is researched. If two users like the same games, the system try to find the similar items according to this and then recommend the appropriate items, as shown in Fig. 3.

4) Content-based filtering

Content-based recommendation technique makes its suggestions relies on descriptions and attributes about the items and uses the user's interests and preferences (the users predefined profile) to make the recommendations personal [3].

5) Hybrid techniques

As it is in its name, hybrid recommendation system uses multiple recommendation techniques to create more accurate recommendations and personalized suggestions to the users.

B. Similarity measurement techniques

Talking about recommendation systems, measuring the similarity is a key point in the systems. Even if we talk about collaborative filtering or content-based systems, someway the similarity should be determined to find the patterns between the users or items.

For measuring similarity there are many techniques and equation to define it between items. The two most popular similarity measures used for recommendation systems are the cosine similarity and the Pearson similarity.

1) Cosine similarity: Items are corresponding for vectors of an n-dimendional space and their similarity is the cosine of their angle [2].

$$\cos(x, y) = \frac{(x \bullet y)}{||x|| ||y||} \quad (1)$$

where $\bullet$ indicates vector dot product and $||x||$ is the norm of vector x.

2) Pearson similarity: Use the correlation between the items to give the similarity between two objects [2].

$$Pearson(x, y) = \frac{\sum(x,y)}{\sigma_x \times \sigma_y} \quad (2)$$

where the covariance of data points x and y $\sum$ and their standard deviation σ.

III. Development and Implementation

For developing the board game recommendation system three main steps were determined. Talking about a Big Data project, the first step is collecting data about board games. After the data is stored to files, the next step is the data cleaning and preparation phase. The last step demonstrate in this article is the building of the collaborative filtering recommendation systems.

Python programming language was used to write the code, and a few useful libraries to the given steps. These libraries will be discussed and introduced later in the relevant chapters. For backup and version controlling Git and GitHub were used to commit the changes in the program and store the code in GitHub. For writing the relevant code the Visual Studio Code open-source code editor was used.

A. Collect the data

As it can be seen in the previous chapter creating a project under the scope of Big Data need a huge amount of data. However, owning the required quantity and quality of data is not always an easy task.

Data can be collected from the following places:

- from databases of different systems (e.g., database of enterprise software)
- from survey forms filled by users
- from Internet, using a script to scrape interesting data from a given website – web scraping technique

In the present work the web scraping technique was used. For this purpose, the playwright and the beautiful soup python-based libraries were used [5].

Web scraping is a method to collect data without any connection with an API or without a user's interaction with the collectible data. The basis of web scraping is creating an automated program which will ask the web server, using queries to the needed data.

The collection of the data was scraping goes 100% from BoardGameGeeks.com [1].

Three types of datasets were created from the gathered values:

1) Board games general data: Gathered data as a table format from collection pages, where all the games – existing in the website's database – are displayed. Here the most important attributes were like game's title, the ranking order, the general informations about ratings (number of ratings and the average ratings) and the unique link to the game which later will be the identifier of games in all datasets, as can be seen in Fig. 4.

link_to_game	rank	title	avg_rating	number_voters
/boardgame/174430/gloomhaven	1.0	Gloomhaven	8.71	51212.0
/boardgame/161936/pandemic-legacy-season-1	2.0	Pandemic Legacy: Season 1	8.58	47092.0
/boardgame/224517/brass-birmingham	3.0	Brass: Birmingham	8.66	29491.0
/boardgame/291457/gloomhaven-jaws-lion	4.0	Gloomhaven: Jaws of the Lion	8.61	21080.0
/boardgame/167791/terraforming-mars	5.0	Terraforming Mars	8.40	80129.0

Fig. 4. Part of a general dataset about board games' main properties.

2) Board games detailed data: Using the unique identifier link for the games, it is possible to reach more information about board games to reference this. For collaborative filtering this dataset will not be used in the later steps.

3) Ratings data: The user's ratings to the games can be gathered also with the help of the game's unique identifier, which is a part of the whole hyperlink leads to the board games. For collecting ratings a simple schema was built for the file, where the data is stored. As can be seen in Fig. 5, the schema contains the user's username in the website, the rating which they gave to the particular games and the identifier to the games.

	user	rating	game_id
0	annecoleens	8.0	/boardgame/174430/gloomhaven
1	gvaldizan	8.0	/boardgame/174430/gloomhaven
2	Patryk_Purczynski	8.0	/boardgame/174430/gloomhaven
3	hankAA	8.7	/boardgame/174430/gloomhaven
4	Crash G	6.0	/boardgame/174430/gloomhaven

Fig. 5. Ratings dataset's structure, and the collected properties.

Playwright library was used in web scraping process for requesting HTML pages from web server and for browser automation. The other important extension library is Beautiful Soup, this one was responsible for detecting the HTML tags storing the data and scraping this information into python data structures. For collecting data from the website another library Pandas was used as well for piling up data, stored in table format on the site. Finally, the gathered data was saved into CSV files.

B. Cleaning the data

After the data is collected, it is obvious that for using them for the main purposes, some transformation and preparation is essential before start operating with them.

Data cleaning is an integral part of data preprocessing. Data preprocessing is a well-defined process which give back from given data the same- or less amount of data. The essence of data cleansing to detect and remove the errors and inconsistencies occur in data, in the interest of increasing the quality of data [6].

In this step knowledge about our own data and make some analysis on them is important. The better the data is prepared; the more accurate results can be expected from the system in the future.

In this project the used libraries for the data preprocessing were the Pandas and Seaborn python-based libraries. The transformation and cleansing steps were made with Pandas, and the Seaborn was reliable for making charts and plots for further analysis.

Preparing boardgame and ratings datasets for recommendation system, the below steps were taken:

- remove unimportant attributes from datasets
- rename the attributes to bring them to the same naming convention
- detect and drop duplicated rows
- set the index columns for the data frames
- transform inconsistent data to missing values or drop them
- delete board games which would not be recommended by the recommendation system (remove board game extensions from dataset)
- drop rows from ratings dataset where the username or the rating are missing (in that case the row is irrelevant from the point of view of recommendation system's purpose)
- remove the whitespaces (trim) from all values and set the data types to the given columns

Analysis mainly created for ratings dataset to determine some interesting facts on the data, as illustrated in Fig. 6. and Fig. 7.

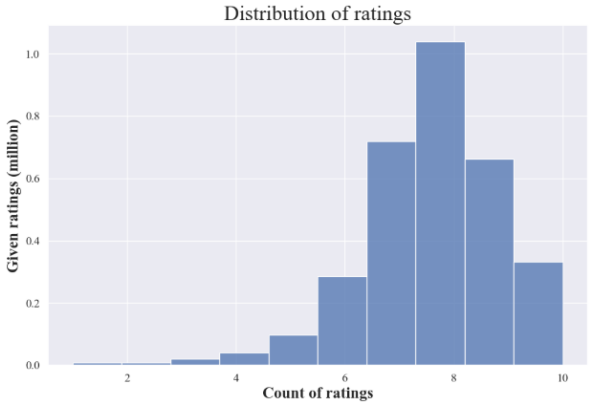

Fig. 6. The distribution of ratings.The most given ratings to the games was between the range of [6.5;9] intervals.

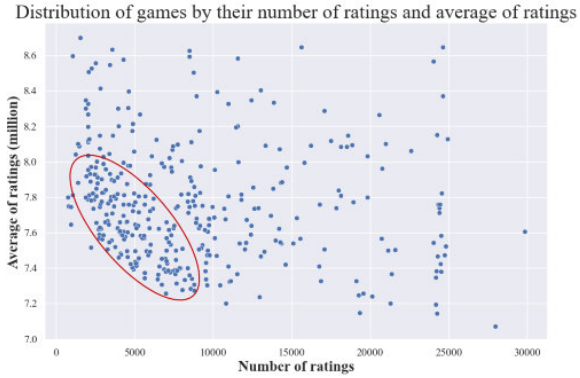

Fig. 7. Distribution of games by their number of ratings and average of ratings. Within the red oval there are the most games appearing.

User-item matrices were generated from ratings dataset to the user-based and item-based recommendation systems as well. Because of the size of the ratings data is so enormous, some processes were executed on the created matrices.

The following attempts were fulfilled to make the appropriate matrices:

- drop users whose number of ratings are below a given threshold number
- drop ratings which are below a given rating threshold, it was determined by using the scale of the ratings and chose an appropriate value which above the user rather likes the game than not
- reduce ratings per game up to a threshold value

The above transformations are required for optimizing the dataset to be able to process on the computer with lower capacity, used for developing this project. At the end of the preparation and transformation of the data for creating, the dataset is counting 370 games with 24570 users and their ratings.

C. *Build the recommendation systems*

For creating a system, resulting satisfying results, more recommendation engines need to be developed with different types of parameterizations and implemented logic, to determine which variations are the best.

The Pandas, Scikit-learn and the SciPy python libraries were used for creating the user-item matrices, to use and execute the Pearson similarity and cosine similarity.

Developing the following recommendation engines, the logic from [7] and [8] were implemented, optimized, and customized to board game ratings dataset.

The general steps in user-based and item-based recommendation algorithms are slightly differ from each other, as can be seen below.

General Steps of user-based recommendation algorithm:

1. Find similar users based on how they interacted with the common items.
2. Find the items which are rated high by both the similar users and the user to whom the recommendation is made. Filtering the cases the rated movies are same between both sides.
3. Calculate the weighted average score for all of the items.
4. Calculate the rank of the items, using the score of the games. Finally pick the top n recommendations [7].

General Steps of item-based recommendation algorithm:

1. Using the user ratings dataset calculate the item similarity scores.
2. Find the top n items which have the highest similarities for the rated and liked items by the user.
3. Calculate the weighted average score for the most similar items by the users.
4. Rank items based on the scores and pick top n items to recommend [8].

For the above algorithms the data needs to be standardized to bring the values to the same interval. The created engines for identifying similarity used the cosine similarity or the Pearson similarity. In the case of cosine similarity, the missing values were replaced with 0 values. The matrices new values will be numbers between [-1;1] intervals, where the closer the number to one the bigger the similarity is between elements. Fig. 8. shows an example for similarity matrix.

user user	-mIDE-	...Hammer	0010	0492372665
-mIDE-	1.000000	0.866025	NaN	-0.612372
...Hammer	0.866025	1.000000	0.000000	0.394719
0010	NaN	0.000000	1.000000	NaN
0492372665	-0.612372	0.394719	NaN	1.000000

Fig. 8. User similarity matrix with Pearson correlation for the first 4 users.

After this process, in the case of user-based collaborative filtering, the test user's ratings were removed from the similarity matrix's index. Setting the threshold to filter out the positive similarity between the users, then sort the similarity values from highest to lowest. Next step to create a matrix already does not contain the boardgames rated by the test user and only contains items rated by the similar users. The following process is the recommendation, where the recommended items are determined by the weighted average of user similarity score and the boardgame rating.

The final similarity weighted rank for the boardgames was calculated by (3):

$$game\ rank = \frac{\sum user\ similarity \times game\ rating}{all\ ratings} \tag{3}$$

where *user similarity* is the similarity measure between the target user and the similar user, *game rating* is the rating given by the similar user and *all ratings* is the number of rated games by all the similar users. Fig. 9. contains the code snippet for implementation of user-based algorithm.

```
ratings_UCF = ratings_UCF \
  .pivot_table(index='user', columns='Title', values='rating')
ratings_UCF = ratings_UCF \
  .subtract(ratings_UCF.mean(axis=1), axis = 'rows')
user_similarity_p = ratings_UCF.T.corr()
user_similarity_p.drop(index=test_user_id, inplace=True)
similar_users = \
  user_similarity_p[user_similarity_p \
    [test_user_id]>similarity_threshold] \
  [test_user_id].sort_values(ascending=False)[:topN]
rated_games_by_test_user = \
  ratings_UCF[ratings_UCF.index==test_user_id] \
    .dropna(axis=1,how='all')
rated_by_similar_users = \
  ratings_UCF[ratings_UCF.index.isin(similar_users.index)] \
    .dropna(axis=1, how='all')

similar_games = rated_by_similar_users.columns
for game in similar_games:
  movie_rating = rated_by_similar_users[game]
  total_scores = 0
  count_num_of_scores = 0
  for user in similar_users.index:
    if not pd.isna(movie_rating[user]):
      score = similar_users[user] * movie_rating[user]
      total_scores = total_scores + score
      count_num_of_scores = count_num_of_scores + 1
  recommended_item_score[game] = \
    total_scores / count_num_of_scores
```

Fig. 9. Code snippet of the user-based collaborative filtering recommendation algorithm. Used similarity measure is Pearson correlation.

In the case of item-based collaborative filtering the steps of the algorithm are changed in some point. In this algorithm after calculating the similarity between the items, selecting the set of games which were rated by the target user. In the same way, the set of games which the target user did not rated are also defined. The next step is predicting the similarity between the rated games and the not rated games. This correlation between items is calculating by the average of ratings and the similarity scores (how the items are similar) using weights. After that we sort the values from highest to lowest and pick the first top N items, which are the most like the target user's rated games.

Equation (4) used for calculating the average of ranks using similarity scores as weights:

$$game\ rank = \frac{\sum rating\ of\ game \times similarity\ score}{all\ ratings} \quad (4)$$

where *rating of game* is the ratings for boardgame given by target user, *similarity score* is the similarity measure between the rated and not rated games and *all ratings* is the number of rated games by the target user. The relevant code snippet can be found in Fig. 10.

```
ratings_ICF = ratings_ICF \
    .pivot_table(index='Title', columns='user', values='rating')
ratings_ICF = ratings_ICF \
    .subtract(ratings_ICF.mean(axis=1), axis = 0)
item_similarity_c = \
    pd.DataFrame(data=cosine_similarity(ratings_ICF.fillna(0)), \
    index=ratings_ICF.index, columns=ratings_ICF.index)
non_rated_games_by_test_user = \
    pd.DataFrame(ratings_ICF[test_user_id].isna()).reset_index()
non_rated_games_by_test_user = non_rated_games_by_test_user
non_rated_games_by_test_user = \
    non_rated_games_by_test_user[non_rated_games_by_test_user \
        [test_user_id]==True]['Title'].values.tolist()
rated_games_by_test_user = pd.DataFrame(ratings_ICF[test_user_id] \
    .dropna(axis=0, how='all').sort_values(ascending=False)) \
    .reset_index().rename(columns={1:'rating'})

for non_rated_game in non_rated_games_by_test_user:
    similarity_score_with_non_rated_game = \
        item_similarity_c[[non_rated_game]].reset_index()\
            .rename(columns={non_rated_game:'similarity_score'})
    similarity_with_test_user_rated_games = \
        pd.merge(left=rated_games_by_test_user, \
        right=similarity_score_with_non_rated_game, \
        on='Title', how='inner') \
        .sort_values('similarity_score', ascending=False)[:topN]
    predicted_rating = round(np.average(
        similarity_with_test_user_rated_games[test_user_id],
        weights=similarity_with_test_user_rated_games \
        ['similarity_score']), 6)
    rating_prediction[non_rated_game] = predicted_rating
```

Fig. 10. Code snippet of the item-based collaborative filtering recommendation algorithm. Used similarity measure is cosine similarity.

IV. Testing and Results

During development occurred challenges, needed to solve and test to find out the most appropriate version of the investigated terms. Furthermore, it was necessary to test how the prepared recommendation systems perform and which one has the most accurate suggestions.

A. Optimize ratings dataset

The size of the dataset containing the scraped games, the users, and the given ratings to games by the users occurred memory problems during processing it with the recommendation algorithm. The error was occurred during the calculations of the similarity matrix.

For optimizing these problems some limitations were introduced in data cleansing process to reduce the size of the file. In addition, after data was loaded for the recommendation phase the following further steps were taken:

- data types were converted to type takes up less memory
- introduced garbage collector to delete non-used variables, thereby freeing up memory

For determine the used memory by the file a Pandas function was used. The main aim was reaching the limit memory usage be around 35-40 megabytes which can still be handled by the computer performing the operation. At the beginning the size of the file which will be using for the similarity matrices was around 123 megabytes, but after the transformations and preparations theses measure only counts 39 megabytes which can be handled by the computer already, without throwing any error.

Another solution would be for this problem is using Spark or Map Reduce technology or acquiring a more powerful computer for higher performance calculations.

B. Test and evaluate recommendation systems

In order to see how the recommendation systems perform testing, the evaluating are important steps. In present work the created systems are working with offline data, which means not real-time user interactions and ratings are the basis of the recommendations, instead pre-saved historical data was scraped from the website.

Big disadvantages of offline dataset are that hard to evaluate them, because of the lack of user interactions. Thus, the recommender's influence on user behavior cannot be directly measured.

As the goal of the offline evaluation is to filter algorithms, the data used for the offline evaluation should match as closely as possible the data the designer expects the recommender system to face when running it online [2].

In the present research work picked test users will be determined than the games which was rated by them will be compared with the recommended items. For measuring accuracy, the Mean Absolute Error and Root Mean Squared Error will be used [2]. Accuracy is one of the most discussed properties measuring while testing recommendation systems, but there are simple equations to work with.

For testing the systems according to what results it gets one user is selected and recommend him or her items with the systems. For this test case user *buffobass* was chosen. This user's 29 ratings can be found in the dataset, the system is working with. It can be said about this user that likes boardgames which genres are strategy, family, party or thematic. It is also known that he or she rather likes the more complex games with longer playing time. The preferred mechanisms in the games the card/drafting games, hand management games, set collection, dice rolling and cooperative games (there are many preferable, but these are the most frequents). The first 20 games rated by the test user can be seen in Fig. 11. with some comments to highlight the common properties.

Title	Comment
Kemet	Strategy, Hand management, Drafting
Horrified	Thematic, Family, Cooperative
Cosmic Encounter	Strategy, Thematic, Hand management
Castles of Mad King Ludwig	Strategy, time 90 minutes
Ora et Labora	Strategy, time 60-180 minutes
Ticket to Ride: Nordic Countries	Family, Hand management, Drafting
Captain Sonar	Thematic, Party
Legendary: A Marvel Deck Building Game	Thematic, Card game
T.I.M.E Stories	Thematic, Dice rolling, Cooperative
Secret Hitler	Party, Card game
Deception: Murder in Hong Kong	Party
My City	Strategy, Family
Fantasy Realms	Family, Card game, Drafting, Set collection
KLASK	Family, Party
Onitama	Abstract, Hand management
Exit: The Game – The Abandoned Cabin	Thematic, Cooperative
Century: Spice Road	Family, Card game, Hand management
Bärenpark	Family, Drafting, Set collection
Awkward Guests	Thematic, Family, Card game, Hand management
Meadow	Strategy, Family, Hand management, Drafting

Fig. 11. The first 20 games rated by the test user.

Investigating the recommended items, it can be said, that the suggested games for the most part have the same genres as the rated games by the test user. The recommended items have also higher playing time, with complex rules and with similar mechanics what the test user prefers. It is also an interesting connection that because of the test user liked one game with content of trains therefore 3 out of 4 recommendation systems recommended games with the same train/railways content, with similar mechanisms and genres.

Fig. 12. shows the recommendations of the item-based recommendation system made with cosine similarity, where some comments were added, to describe the main common properties of the games.

Item-based with cosine	Comment
Commands & Colors: Ancients	Dice Rolling, Hand Management
Agricola: All Creatures Big and Small	Strategy
For Sale	Family, Hand management, Card game
Samurai	Strategy, Hand management, Set collection
Flamme Rouge	Family, Hand management
Dune	Thematic, Strategy, time 120-180 minutes
Railways of the World	Strategy, time 120 minutes
Ticket to Ride	Family, Hand management
Memoir '44	Dice rolling
1960: The Making of the President	Strategy, Hand management

Fig. 12. Top 10 recommendation with item-based recommendation system with cosine similarity.

By using the different type of recommendation systems, it can be observed that even changing only the similarity parametrization the results will be differ from each other, but the recommended items fit to expectations.

V. Conclusion

This work presents different types of recommender systems that have been developed for recommending board games. As it can be seen in the presented test case the created recommendation systems based on item-based collaborative filtering are able to recommend items which suits the user's preferences.

The other types investigated recommendation systems have coverage with these suggestions, but mostly the recommended games vary from system to system. But it is clear, that the resulted items own very similar properties in both the rated games by the test user and the recommended ones.

Introducing more testing and evaluation techniques and by drawing conclusions from the results more accurate and user-friendly recommendation systems may be built.

References

[1] „Board Game Geeks," Available: https://boardgamegeek.com/. accessed 23 September 2022.

[2] F. Ricci, L. Rokach, B. Shapira és P. B. Kantor, Recommender Systems Handbook, New York: Springer, 2011.

[3] L. Sharma és A. Gera, „A Survey of Recommendation System: Research Challenges," International Journal of Engineering Trends and Technology, volume 4, issue 5, pp. 1989-1992, 2013.

[4] M. S. P. Babu és B. R. S. Kumar, „An Implementation of the User-based Collaborative Filtering Algorithm," International Journal of Computer Science and Information Technologies, volume 2, issue 3, pp. 1283-1286, 2011.

[5] R. Mitchell, Web Scraping with Python, Sebastopol: O'Really Media, 2018.

[6] J. Han, M. Kamber és J. Pei, Data Mining: Concepts and Techniques, Cambridge: Elseiver Science & Technology, 2011.

[7] Amy GrabNGoInfo, „Medium," Available: https://medium.com/grabngoinfo/recommendation-system-user-based-collaborative-filtering-a2e76e3e15c4. accessed 25 September 2022.

[8] Amy GrabNGoInfo, „Medium," Available: https://medium.com/grabngoinfo/recommendation-system-item-based-collaborative-filtering-f5078504996a. accessed 25 September 2022.

[9] M. Qutbuddin, „Towards Data Science; An Exhaustive List of Methods to Evaluate Recommender Systems," 20 April 2020. Available: https://towardsdatascience.com/an-exhaustive-list-of-methods-to-evaluate-recommender-systems-a70c05e121de. accessed 28 September 2022.