This is the accepted manuscript of the article, which has been published in Flouris, Giorgos et al. (eds) Information Search, Integration, and Personalization : 13th International Workshop, ISIP 2019, Heraklion, Greece, May 9–10, 2019, Revised Selected Papers. ISBN: 978-3-030-44900-1.
https://doi.org/10.1007/978-3-030-44900-1_9

A Hybrid Recommender System for Steam Games

Jin Gong, Yizhou Ye, and Kostas Stefanidis

Tampere University, Finland
{jin.gong,yizhou.ye,konstantinos.stefanidis}@tuni.fi

Abstract. A recommender system can be considered as an information filtering system that seeks to predict the preference a user would have for a data item. It is commonly utilized in digital stores to recommend products to their users according to the users previous purchases. This applies to Steam as well, a widely used digital distribution platform for games. The existing recommender system mainly suggests new games to a given user by calculating similarities between games they own and those that they do not. These similarities are based on predefined attributes (game genres). Additionally, the system is able to recommend games based on the game preferences of the users friends. In this work, we target at creating an enhanced recommender system for Steam. The goal is to design a hybrid approach for producing suggestions that will utilize data, such as playing time, game price and game release date, in addition to the genres and the preferences of friends.

Keywords: Steam; User Profile; Recommendation System; Collaborative Filtering

1 Introduction

The Steam platform is the largest digital distribution platform for PC gaming nowadays. On 14 Jan 2019, Steam published its annual report based on the past 12 months, including data on stores, the Steam community, gameplay, Steamworks, and things behind the scenes. According to the report, Steam users have experienced explosive growth in 2018 [18]. Among them, the daily active users of the platform are up to 47 million, the monthly active users are 90 million, the highest number of online users is 18.5 million, and the monthly growth of users with valid purchases is 1.6 million. One of the key reasons Steam is growing so rapidly is the good search-ability in store, which was mentioned in the report as well. They are working on a new recommendation system driven by machine learning to find games that match the player's personal preferences. Although the algorithm is just part of the search-ability solution, they are also building more live and appreciation features and continually evaluating the overall design of the store.

On the other hand, the recommender system sometimes does not work as well as expected. One reason for this is the *Matthew Effect* [15], which means

the rich gets richer and the poor gets poorer always appears in the social science field, can be also applied to the game market. Those games developed by big game developing companies receive more budget on advertisement so that they can be very popular. Popular games will appear on the top position on the store web pages and attract more users to buy including you and your friends. Meanwhile, games developed by small studios or individual developers are not that lucky. Those budget games without enough attention would easily disappear from users. What is even worse, if developers are unable to earn money from those games, they are very likely to break down which does harm to the whole game market. This is the reason why people need a recommender system, for suggesting niche games to users.

Our goal, in this paper, is to create a hybrid approach for producing suggestions that will utilize data, such as playing time, game price and game release date, in addition to the genres and preferences of friends used already. Our initial target is to analyze the data, which comes in three parts. The first part includes the user IDs, the games that they purchased and the hours they had spent playing the game[1]. The second part consists of the game titles combined with their prices and release dates[2], while the third part consists of the game names, game IDs and their genres. This dataset was manually crawled from the Steam API.

We convert all available data into numerical ratings ranging from 1 to 5. These ratings will then be used in calculating the Pearson correlation between the cases to determine the similarities. A rating of 1 equals to total similarity and -1 means that the entities being compared are total opposites to each other. For producing recommendations, the final rating of a game to any given user is the mean of several values, including user preference of this game genre, user preference of games similar to this game, similar users preference of this game, user preference of price and user preference of game released in that time zone. The proposed approach includes no assigned weights to the individual parts for computing the overall ratings, i.e., each aspect of the data would have an equal amount of impact on the final rating. Python is used as the programming language of the system.

We evaluate the accuracy of our results by separating the dataset into a test set and a training set. Specifically, we attempt to predict the values in the test set using the training set.

This paper is organized as follows. Section 2 discusses about the recommender system in use of the Steam platform and the recommender of other platforms. Section 3 introduces the dataset and analysis method. Section 4 shows the performance of the proposed approach, and Section 5 concludes the paper with a summary of our contributions.

[1] https://www.kaggle.com/tamber/steam-video-games/
[2] https://www.kaggle.com/ kingburrito666/over-13000-steam-games/

2 Related Work

2.1 Recommender System of Steam

Valve Software[3] is keeping improve the recommending system of Steam platform annually. However, Valve has never gave out any detailed information about the algorithm of its recommender system. The system is like a black box to common users. Some analysts (e.g., Erik Johnson) have attempted to dive deeply into the mechanism behind Steam. Erik Johnson stopped using his personal Steam account and spent two months recording all the games he had played, viewed and commented since then, as well as the play-time, and he even stored the HTML pages of those games. During the two months, he viewed all the 672 games recommended by Steam and started to find the relationships among them.

According to this study, Eric Johnson found that the system could make better recommendations if it relied less on popularity and recency, and instead did a better job of surfacing titles based on quality and personal relevance factors. The challenge here is that popularity and recency are easy to quantify. Quality and relevance are more elusive. Furthermore, the most surprising omission in all these systems is the lack of collaborative filtering [10].

2.2 Other Works on Recommender Systems

The recommender system is not a unique feature of the game platform. Since the popularity of the Internet, various platforms have used their unique recommender systems to provide customized services to users.

In general, a recommender system aims at providing suggestions to users or groups of users by estimating their item preferences and recommending those items featuring the maximal predicted preference. Typically, depending on the type of the input data, i.e., user behavior, contextual information, item/user similarity, recommendation approaches are classified as content-based [19], collaborative filtering [20], knowledge-based [4], hybrid [2], or even social ones [22]. Nowadays, recommendations have more broad applications, beyond products, like links (friends) recommendations [28], query recommendations [6], health-related recommendations [23, 24], open source software recommendations [11], diverse venue recommendations [7], recommendations for groups [14, 16, 17], sequential recommendations [25, 3] or even recommendations for evolution measures [21, 27].

Next, we take as an example a shopping experience to showcase how recommenders work. So, typically, recommender algorithms start by finding a set of customers who purchased and rated items overlap the users purchased and rated items. The algorithm aggregates items from these similar customers, eliminate items the user has already purchased or rated, and recommends the remaining items to the user. In the case of item-to-item collaborative filtering, like in our work, the focus is on finding similar items, not similar customers. For each of the

[3] https://www.valvesoftware.com/en/

users purchased and rated items, the algorithm attempts to find similar items. It then aggregates similar items and recommends them (e.g., [13]). This is the method that many companies, like Amazon, are using. However, this method is more likely to be a *have-to* choice due to the lack of *friends* feature, so its recommendations rely on *items* and similar users. It would guess which product you may like according to your wish list, your previous purchase and the goods you searched for.

There are also shopping websites which allow users to add each other as *friends* and take what your friends bought into account. Some people may have similar experience like: *The shop recommends the item I just bought.* Here we are going to introduce a concept that is the cost of making mistakes. A bad recommendation will have a bad effect, but the question is how big the effect. The cost is very small in shopping field because when a user opens the website, he/she knows clearly what he/she needs. If they recommend him/her something, he/she does not need, this user will even never click on it. The game field is kind of the opposite. Many users open the game store without a specific game need. They just select a game for fun, no matter which one. It is very difficult to judge a game according to its description and several images. For example, based on the description and images, the user might think that he/she likes it, but after trying the game, he/she disliked it. During this period, the user needs to pay both money and time, leading to trust reduction of the user for the system.

This is totally different in another field, like YouTube. The top sector on YouTube web page is the recommendation of the video. Its algorithm is based on the channels you subscribed, the videos you previously viewed and videos you liked. They use two neural networks. The first one is the candidate generation network, it takes events from the users YouTube activity history as input and retrieves a small subset (hundreds) of videos from a large corpus. The second one, the banking network, it accomplishes this task by assigning a score to each video according to the desired objective function, using a rich set of features describing the video and user. The highest scoring videos are presented to the user, ranked by their score [5]. For new user without any interactions with the system, the system will show the most trending videos based on the user's location. This mechanism is more like the Steam platform. It will classify users with many tags. For instance, if you have viewed many technology videos, you may gain a tag says *technology fans* and you may get as well other tags like *nature fans* or *fans of a pop star*, if you watch videos of that type. The steam platform is doing the same thing as well. If you played a lot of free games, they are highly likely to introduce other free game rather than those very expensive fee-paying ones to you.

3 The Dataset

To do deep analysis with Steam users gameplay, a sufficient amount of data is needed. For our analysis, 100,000 users will be used. The open-source data company with educational datasets should be the best choice, namely, Kaggle.com.

Table 1: Dataset information.

Type	Size	Description
Users	11350	The number of users
Games	5155	The number of games
Games per user	17.62	The number of games owned by a user in average
Gaming time (in hours)	0.1-11754	The time a user spends on a game
Year of publish	2007-2017	The year the game published in

According to the Terms and Use of Kaggle.com, Steam's dataset can be downloaded for academic use, obeying any ethics issues. We found three different datasets from Kaggle. The first one [26] includes user IDs, the games that they purchased and the hours they had spent playing the game. The second dataset [12] includes the game titles combined with their prices and release dates. The third one only includes the game titles initially, however, for the hybrid recommendation system that we are aiming to create, we want to add the game genres into the mix.

A web crawler based on Python language was created to collect game genres from public game profiles. Steam URLs are of the form /gameID/gameName, and we had no access to gameIDs in our data. After realized that using the Steam webpage would be far too inconvenient due to game name in the URL being inconsistent with the name provided in the data, the next attempt was made using a website containing information on all Steam games, SteamDB. Much to our dismay, only trials and errors are learned because this website actively blocks all crawling scripts, although we did manage to find the source that this website is using and ended up using that instead[1]. The gameIDs along with the game genres would then be added to the third dataset. However, many games (2030 cases) have missing genres due to several reasons: Some of them were old and thus removed from the current Steam store. Others could not be found because their names were spelled differently in the sources that we were comparing.

Table 1 shows the basic information about the dataset. If the whole dataset is changed into a user-game table, the known data (the user-game pair) only fills 0.34% of the whole table. The data itself is very sparse so any analysis based on the raw data is inefficient and inaccurate. Features should be extracted from the dataset for more research.

4 The Method

All available datasets are in a csv-format. Python libraries, like Numpy and Pandas, are used to organize the data. In the first dataset, there was an empty column that we removed. All the cases with missing genres are excluded.

Next, we transform the initial three datasets into four different tables.

Table 1: User-Game Table. We will first create the user-game table, which is a table containing users along with their ratings for the games based on the time

they had spent playing the game. Our rating system is an interval from 1 to 5, and it was calculated by dividing the playing time into 5 equally parts [29]. For example, a given user would get a rating of 5 for a given game if the time the user had spent playing the game belonged in the top-20% out of all the users.

Table 2: User-Genre Table. We will then create the user-genre table by taking the average of all the ratings for games that belong in the same genre [8]. For example, let us assume that a user has game1 and game2. Game1 belongs to genre1 and genre2, game2 belongs to genre2. The rating this user would obtain for genre1 would be the rating they got for game1. Likewise, the score for genre2 would be the mean of game1 and game2. The formula can be generalized as follows:

$$score(user\_i, genre\_j) = \frac{\sum\limits_{g \in G_n} game_g\_rating}{n} \tag{1}$$

where n equals the number of games that $user_i$ has in $genre_j$, and G_n is the set of n games that $user_i$ has in $genre_j$.

Table 3: User-Price Table. We started by defining three price ranges:

- Free to Play games.
- Games that are below 20$, but not free.
- Games that are above or equal to 20$.

First, we calculate the number of games that a user has in each price range. The rating for that price range would then be the mean of all the ratings for games in that price range, e.g., if a given user had 4 games, game1(free), game2($<$20$), game3($\geq$20$) and game4($\geq$20$), then the rating this user has for free games would be the rating for game1, the rating for games under 20$ would be the rating for game2 and the rating for games over 20$ would be the average of $rating\_game3$ and $rating\_game4$.

Table 4: User-Release Year Table. It should be noted that the data we had only had release dates ranging from 2007 to 2017, which may cause some bias in our results, though Steam does remove older titles from the store regularly. They remain playable if the user had purchased them but cannot be bought from the store anymore. We categorized these release dates into three categories:

- Older than 2010.
- Games released between 2010 and 2015.
- Newer than 2015.

In this case, the rating system works exactly like in the previous user-price table, which is to say, the rating for a certain interval is the mean of all the ratings the user had given for games that belong in that interval.

Overall Aggregation. Pearson correlation is used to measure the similarities between the users and the similarities between the games. A score of 1 would equal total similarity and -1 would mean that the entities being compared are total opposite of each other. The Pearson correlation similarity of two users x, y is defined as:

$$simil(x,y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r}_x)(r_{y,i} - \bar{r}_y)}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r}_x)^2} \sqrt{\sum_{i \in I_{xy}} (r_{y,i} - \bar{r}_y)^2}} \tag{2}$$

where I_{xy} is the set of items rated by both user x and user y, $r_{a,b}$ the rating assigned to game b by user a, and $\bar{r}_a$ the mean of the ratings for user a.

After obtaining the similarity matrices, the system is able to produce recommendations of similar users for any given user. The same is true for games. This is accomplished using the prediction formula which is the same as what we use to calculate the correlation of two users. But the explanation should be the Pearson correlation similarity of two items x, y is defined as that, and where I_{xy} is the set of users give rate to both game x and game y.

Next, let us explain the system with a practical example: Bob does not own the game Dota2. Dota2 belongs to several genres: Action, Free to Play and Strategy. As is apparent from the genres, it is free. It was released in 2013. The final rating that Bob receives for Dota2 is the mean of five different ratings:

- The mean rating for Dota2 given by top-5 most similar users.
- The mean rating is given by Bob for top-5 most similar games.
- The rating Bob gave for games released between 2010 and 2015.
- The mean rating Bob gave for Action, Free to Play and Strategy-games.
- The rating Bob gave for games that are Free to Play.

5 Experiments

We evaluated the accuracy of our results by randomly selecting 10% of the data as the test set. The remaining 90% of the data would be used as the training set. We would randomly select one rating from a user and delete it. Subsequently, we would try to predict the deleted value using the training set. These predictions were then analyzed through MAE and NMAE [9]:

$$MAE = \frac{\sum_{i \in N} |p_i - q_i|}{n} \tag{3}$$

$$NMAE = \frac{MAE}{R_{max} - R_{min}} \tag{4}$$

50 independent tests with random seeds have been done to select the different testing set and training set. The testing results were transformed into charts for better understanding (see, Figures 1 and 2).

Fig. 1: MAE scores in 50 iterations.

It can be seen from the charts, the average of MAE is around 6 which stands for the average of prediction is about 2.45, and the average of NMAE is around 0.33 which means our model has an accuracy around 67% on predicting. Which means, every time the user is viewing a video game product, this system will automatically recommend five other video games according to the current game and the preference of the current user. Among the five recommended games, more than 2 of them (actually 2.45 out of 5) will meet the interest of the user. Meanwhile, if the system recommend a game to the user, it is 67% sure that the user will interested in this game or even purchase.

Another test is about how the accuracy change when user and game collaborative filtering, game genre, price and publishing year are considered one by one (see, Figure 3). In this test, the system tries to predict the game that the user likes best or in other words, the games with longest playtime. We randomly select 10% of the data as the test set as well. The remaining 90% of the data would be used as the training set. For all the users in testing set, the playtime information is hidden, we can only know the names of the games they owned, but we do not know how long they spent on the games. The game which a user spent the longest time on is defined as his or hers favourite game. The accuracy is calculated as:

$$accuracy = \frac{N_c}{N},$$

where N_c represent the number of users that predicted correctly and N is the number of users contained in testing set.

If the system recommends games taking into account similarities between users, which in turn means similarities between the games the users own, we calculate similarities using the Pearson correlation (see, Section 4).

With this mechanism (UserFC), the average accuracy is about 13.5%. The second step is take both similar users and similar games into concern. For each

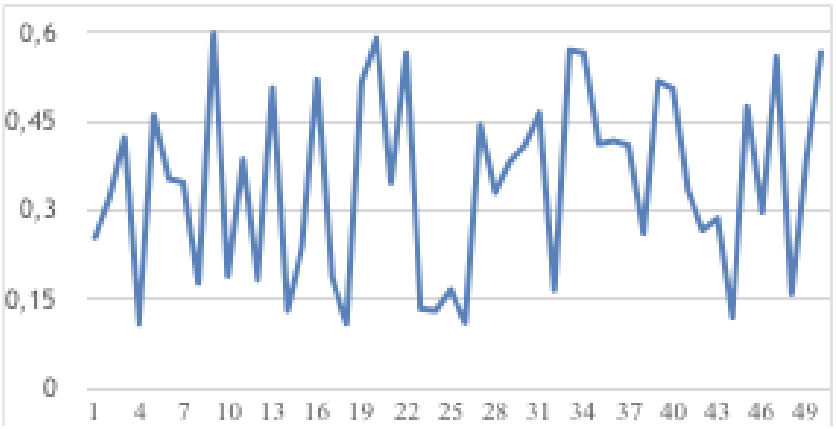

Fig. 2: NMAE scores in 50 tests.

game in the library, calculates the sum of the similarities between the game and all the other games owned by the user. The game with the highest sum score is seen as the game that should be recommended from GameCF. If the UserCF and GameCF give out different games to recommend, just choose the one more popular in all the other users. Then find out the favourite genre, price range and year range of the user, and recommend a game with highest score satisfying one feature above, we proceed as follows. The accuracy of the selected game is just the favourite game of the user going from 13.5% to 22.8%. Among the five factors, the price contribute the most and the publish year is the least important. The result shows that even though from every aspect a game seems that it suits the user very well, but if the price is too expensive, the user will not choose the game.

6 Conclusion

A recommender system can be considered as an information filtering system that seeks to predict the preference a user would have for a data item. In this paper, we focus on the Steam, a widely used digital distribution platform for games. Specifically, we target at producing an enhanced recommender for Steam that uses data, such as playing time, game price and game release date, genres and users preferences for making suggestions.

Based on our first experimental results, we investigate that the big sparsity of our dataset appears to be an important reason for making the results not as good as expected. This is caused by two reasons. The first reason is the raw data itself that is too sparse and cannot be modified. The second one is data loss when different tables are joint together. For instance, a user has 10 games, but we only have information about 5 out of 10. Then, only those 5 games contribute to the analysis, which is still related to data sparsity.

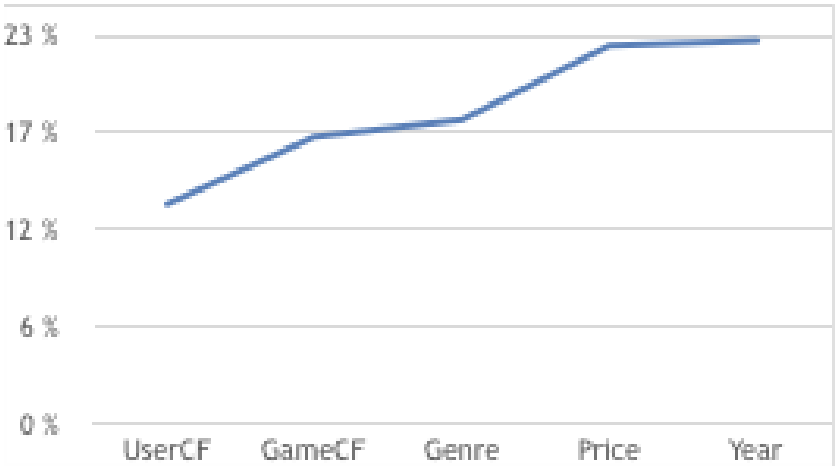

Fig. 3: The accuracy changes.

Another reason is related to the fact, that on the particular domain, it is sometimes random if a user owns a suitable game or not. A game might be very suitable to the taste of one user, no matter what system you use to test, but the fact is that this user does not own the game. He or she might have already had this game on another platform or will meet the game in the future and then own it.

Among the 5 factors listed in the paper, a recommendation based on similar user and similar games seems more reliable which is so-called collaborative filtering. Meanwhile, the price of a game is a factor considered by most of the users tested. The age of the game and the genres seems to be not so important to many users. As we know, the genres are classified by tags contributed by other users and not supervised by Steam platform or game publisher. Misclassification sometimes happens on Steam. It could be a potential reason making genre factor not influential. To generalize, we opt in our future work to assign different weights to different factors, depending on their importance, instead of having all factors that equally effect on the final hybrid rating.

We also tried to generate rules from our results using the IBM's SPSS software tool. Using the Scikit library of Python, we were able to generate some rules, some even with very high accuracy, but the general issue was their low coverage (around 0,01% at best). Thus, we discarded these rules due to overfitting. We leave this as future work, and we will consider in our next steps, alternative ways for generating rules.

References

1. Official team fortress wiki. `https://wiki.teamfortress.com/wiki/`, accessed: Apr. 11, 2019

2. Balabanovic, M., Shoham, Y.: Content-based, collaborative recommendation. Commun. ACM **40**(3), 66–72 (1997)
3. Borges, R., Stefanidis, K.: Enhancing long term fairness in recommendations with variational autoencoders. In: Proceedings of the 11th International Conference on Management of Digital EcoSystems, MEDES 2019 (2019)
4. Bridge, D.G., Göker, M.H., McGinty, L., Smyth, B.: Case-based recommender systems. Knowledge Eng. Review **20**(3), 315–320 (2005)
5. Covington, P., Adams, J., Sargin, E.: Deep neural networks for youtube recommendations. In: Proceedings of the 10th ACM conference on recommender systems. pp. 191–198. ACM (2016)
6. Eirinaki, M., Abraham, S., Polyzotis, N., Shaikh, N.: Querie: Collaborative database exploration. IEEE Trans. Knowl. Data Eng. **26**(7), 1778–1790 (2014)
7. Ge, X., Chrysanthis, P.K., Pelechrinis, K.: MPG: not so random exploration of a city. In: MDM (2016)
8. Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Communications of the ACM **35**(12), 61–71 (1992)
9. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: 2008 Eighth IEEE International Conference on Data Mining. pp. 263–272. Ieee (2008)
10. Johnson, E.: A deep dive into steam's discovery queue. `https://www.gamasutra.com/blogs/ErikJohnson/20190404/340061/A_Deep_Dive_Into_Steams_Discovery_Queue.php`, accessed: Jul. 7, 2019]
11. Koskela, M., Simola, I., Stefanidis, K.: Open source software recommendations using github. In: TPDL (2018)
12. Larson, L.: Over 13,000 steam games. `https://www.kaggle.com/kingburrito666/over-13000-steam-games/`, accessed: Apr. 11, 2019
13. Linden, G., Smith, B., York, J.: Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing (1), 76–80 (2003)
14. Machado, L., Stefanidis, K.: Fair team recommendations for multidisciplinary projects. In: 2019 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2019, Thessaloniki, Greece, October 14-17, 2019. pp. 293–297 (2019)
15. Merton, R.K.: The matthew effect in science: The reward and communication systems of science are considered. Science **159**(3810), 56–63 (1968)
16. Ntoutsi, E., Stefanidis, K., Nørvåg, K., Kriegel, H.: Fast group recommendations by applying user clustering. In: ER (2012)
17. Ntoutsi, E., Stefanidis, K., Rausch, K., Kriegel, H.: Strength Lies in Differences: Diversifying Friends for Recommendations through Subspace Clustering. In: CIKM (2014)
18. O'Neill, M., Vaziripour, E., Wu, J., Zappala, D.: Condensing steam: Distilling the diversity of gamer behavior. In: Proceedings of the 2016 Internet Measurement Conference. pp. 81–95. IMC '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2987443.2987489, `http://doi.acm.org/10.1145/2987443.2987489`
19. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: The Adaptive Web, Methods and Strategies of Web Personalization (2007)
20. Sandvig, J.J., Mobasher, B., Burke, R.D.: A survey of collaborative recommendation and the robustness of model-based algorithms. IEEE Data Eng. Bull. **31**(2), 3–13 (2008)
21. Stefanidis, K., Kondylakis, H., Troullinou, G.: On recommending evolution measures: A human-aware approach. In: 33rd IEEE International Conference on Data

Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017. pp. 1579–1581 (2017)
22. Stefanidis, K., Ntoutsi, E., Kondylakis, H., Velegrakis, Y.: Social-Based Collaborative Filtering, pp. 1–9. Springer New York, New York, NY (2017)
23. Stratigi, M., Kondylakis, H., Stefanidis, K.: Fairness in group recommendations in the health domain. In: ICDE (2017)
24. Stratigi, M., Kondylakis, H., Stefanidis, K.: Fairgrecs: Fair group recommendations by exploiting personal health information. In: DEXA (2018)
25. Stratigi, M., Nummenmaa, J., Pitoura, E., Stefanidis, K.: Fair sequential group recommendations. In: Proceedings of the 35th ACM/SIGAPP Symposium on Applied Computing, SAC 2020 (2020)
26. Tamber: Steam video games. `https://www.kaggle.com/tamber/steam-video-games/`, accessed: Apr. 11, 2019
27. Troullinou, G., Kondylakis, H., Stefanidis, K., Plexousakis, D.: Exploring RDFS kbs using summaries. In: The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I. pp. 268–284 (2018)
28. Yin, Z., Gupta, M., Weninger, T., Han, J.: LINKREC: a unified framework for link recommendation with user attributes and graph structure. In: WWW (2010)
29. Zhang, J., Peng, Q., Sun, S., Liu, C.: Collaborative filtering recommendation algorithm based on user preference derived from item domain features. Physica A: Statistical Mechanics and its Applications **396**, 66–76 (2014)