This is the accepted manuscript of the article, which has been published in New Trends in Databases and Information Systems : ADBIS 2019 Short Papers, Workshops BBIGAP, QAUCA, SemBDM, SIMPDA, M2P, MADEISD, and Doctoral Consortium, Bled, Slovenia, September 8–11, 2019, Proceedings. 2019, 513-524.
https://doi.org/10.1007/978-3-030-30278-8_498_9

GameRecs: Video Games Group Recommendations

Rama Hannula, Aapo Nikkilä, and Kostas Stefanidis

Tampere University, Finland
{rama.hannula,aapo.nikkila,konstantinos.stefanidis}@tuni.fi

Abstract. Video games are a relatively new form of entertainment that has been rapidly gaining popularity in recent years. The number of video games available to users is huge and constantly growing, and thus it can be a daunting task to search for new ones to play. Given that some games are designed to be played together as a group, finding games suitable for the whole group can be even more challenging. To counter this problem, we propose a content-based video game recommender system, GameRecs, which works on open data gathered from Steam, a popular digital distribution platform. GameRecs is capable of producing both user profiles based on Steam's user data, as well as video game recommendations for those profiles. It generates group recommendations by exploiting lists aggregation methods, and focus on providing suggestions that exhibit some diversity by using a k-means clustering-based approach. We have evaluated the usability of GameRecs in terms of the user profile generation and the produced video game recommendations, both for single users and for groups. For group recommendations we compared two recommendation aggregation methods, Borda Count and Least Misery method. For diversity evaluation we compared results with and without the proposed k-means clustering method.

Keywords: Recommendations; Group recommendations; Game recommendations

1 Introduction

Nowadays, video games are a very popular form of entertainment and new games are getting released all the time. Since older games are also still playable, the number of games available to customers is constantly increasing. For this reason, it is not feasible for a user to manually go through every existing game when searching for new games to play. Fortunately, recommender systems can be used to help solve this problem. In general, recommender systems aim at providing suggestions to users or groups of users by estimating their item preferences and recommending those items featuring the maximal predicted preference [10, 11, 2].

Clearly, recommender systems can be also useful in the domain of video games. To our knowledge, the only application for recommender systems to digital games was proposed in [12], in which two different recommender systems

were proposed based on archetypal analysis. However, this work only generates recommendations for a single user at a time. For many multi-player games, it is essential to have a group of friends to play with. Thus, there is need for systems that can recommend games for groups.

In this work, we target at developing a system that could generate diverse and fair game recommendations for groups of users. In other words, the resulting recommendations should contain games that every member of the group would like, but also games that are dissimilar to each other, so as to increase user satisfaction. The system should also recommend both popular games and more obscure games, while prioritizing neither of them.

Specifically, we present a content-based method for recommending games for groups of people to play together. Our method exploits tags that the community has given to games, and using these tags it generates user profiles, and then game recommendations for these profiles. For demonstrating the effectiveness of our approach, we exploit user and game data available at the popular digital distribution platform Steam (http://store.steampowered.com/). We have evaluated the usability of our approach in terms of the user profile generation and the produced video game recommendations, both for single users and groups.

The rest of this paper is structured as follows. Section 2 presents related work, while Section 3 describes the users and games data in GameRecs. Section 4 introduces our approach for group games recommendations. Section 5 presents our usability evaluation results, and finally, Section 6 concludes the paper with a summary of our contributions.

2 Related Work

Recommender systems aim at providing suggestions to users or groups of users by estimating their item preferences and recommending those items featuring the maximal predicted preference. Typically, recommendation approaches can be classified as content-based [10], collaborative filtering [11], and hybrid ones [2]. In content-based approaches, information about the features/content of the items is processed, and the system recommends items with features similar to items a user likes. For example, if a Yelp user is always eating at sushi restaurants, he/she most likely likes this kind of food, so we can recommend him/her restaurants with the same cuisine. In collaborative filtering approaches, we produce interesting suggestions for a user by exploiting the taste of other similar users. For instance, if many users frequently go to Irish pubs after visiting an Italian restaurant, then we can recommend an Irish pub to a user that also shows preference for Italian restaurants. In knowledge-based approaches, users express their requirements, e.g., in terms of recommendation rules, and the system tries to retrieve items that are similar to the specified requirements. Finally, the hybrid recommender systems combine multiple of the aforementioned techniques to identify valuable suggestions.

Nowadays, recommendations have more broad applications, beyond products, like links (friends) recommendations [17], social-based recommendations

[14], query recommendations [5], health-related recommendations [15, 16], open source software recommendations [7], diverse venue recommendations [6], or even recommendations for evolution measures [13]. There is also a lot of work on specific aspects of recommendations due to challenges beyond accuracy [1], like the cold start problem, the long tail problem and the evaluation of the recommended items in terms of a variety of parameters, like surprise, persistence [3] and serendipity [4]. More recently, many approaches that combine numerical ratings with textual reviews, have been proposed (e.g., [8]). For achieving efficiency, there are approaches that build user models for computing recommendations. For example, [9] applies subspace clustering to organize users into clusters and employs these clusters, instead of a linear scan of the database, for making predictions.

Clearly, recommender systems can be also useful in the domain of video games. Due to the large number of game releases every year, gamers can have hard time finding games fitting their interests. To our knowledge, the only application for recommender systems to digital games was proposed in [12]. Two different recommender systems were proposed based on archetypal analysis. Moving forward, in our work, we focus on group recommendations, and on how to identify a diverse set of games to propose to the group.

3 Games and Users Data

In our recommender, we pay attention on data regarding games and users. Specifically, we aim to find out the type of a game according to its tags, and the game type preferences of a user according to the tags of the games that they have played. For doing so, we employ data available at the popular digital distribution platform Steam.

3.1 Games Data

Steam's database contains several types of applications in addition to games, like media editing software and forms of extension packs for games. For example, they have media editing software, which we are not interested in. They also have DLCs (downloadable content), forms of extension packs for games. We are strictly interested in recommending games that can be acquired and played on their own, so we only consider the applications that indicate that their application type is a standalone game.

Each game in Steam has a number of tags. Tags are keywords voted for by the community, and they aim to describe the game with keywords resembling genres, categories, and others. The number of tags can vary greatly. Tags only appear on games if players have voted on them, which leads to popular games often having a greater number of tags than unpopular games. It's possible that a game has no tags at all. A common number of tags for a game would be from 10 to 20. For each game, we know the number of votes of each tag of the game. Thus, a game most likely has more votes on certain tags than others. For

example, some game might have considerably more votes for the "Third-Person Shooter" tag than for the "Crime" tag, in which case we are more interested in the former.

In our recommender, we use the tags and their votes on a game to describe the type of the game. In essence, we consider two games similar to each other if they share many tags with many votes on those tags. From all existing 339 unique tags, we hand-picked 19 tags we did not see fit for describing the type of the game. Some examples of these tags are "Co-op", "Singleplayer" and "Multiplayer", which describe how the game is played instead of what the game is like. We ignore these tags in our recommendation methods.

3.2 Users Data

For users, we are interested in what types of games they prefer. As described above, we determine the type of a game according to their tags and votes. In order to define a user's game preferences, we want to know ratings for tags measuring how much the player enjoys games with those tags. Essentially, we build user profiles containing tag ratings resembling the tag votes on games.

4 Recommendations

In this section, we introduce our method for producing group recommendations for video games. The process is divided into four parts. First, we focus on generating the user profiles, after which we compute single-user recommendations for those profiles. These recommendations are then aggregated into a single list of recommendations and finally, diversified by exploring a clustering method.

4.1 Generating User Profiles

First, our approach generates a user profile for each user. For creating the profile, we exploit the knowledge of which games the user has played and for how long. Each game the user has played for longer than a certain threshold is considered, and for each of these games the user profile's tag rating for that tag is increased. The final user profile contains scaled ratings for tags ranging from 0 to 1, higher value being better. Since some tags are more common than others, we want to prevent very frequent tags from dominating other, less common ones. For this purpose, we use inverse document frequency (IDF), which is calculated as: $IDF(t) = log\frac{N}{n(t)}$, where N is the total number of games in the dataset and $n(t)$ is the number of games that contain the tag t. This means that more frequent tags get lower IDF values than less common tags. IDF is calculated for every tag that appears in any game in the dataset.

To generate the actual user profile, we look at every game in the user's library that the user has played for more than 2 hours. Some kind of playing time threshold is necessary because many users own games that they are not really interested in. It is more safe to assume that users like games they have

actually played for some time. For each of these games, we iterate over every tag t the game g has, and calculate a strength for the tag with the following formula:

$$strength(t, g) = \frac{tagVotes(t, g)}{maxTagVotes(g)} \tag{1}$$

where *strength(t, g)* is the strength value for tag t for game g, *tagVotes(t, g)* is the number of votes the current game g has for tag t, and *maxTagsVotes(g)* is the highest number of votes any tag has for the game g. This results in higher strength for tags that have more votes, with the maximum strength being one and minimum strength being close to zero. It is not possible for a tag to have zero votes, since then the game would not have that tag at all.

For each of these tags that exist in any of the games the user has played, we calculate the strength and add it to the total rating of the tag. The final rating for a tag, called tag rating, is the sum of the tag strengths, multiplied by their respective IDF value squared:

$$tag\_rating(t, p) = IDF(t)^2 \times \sum_{g \in G_p} strength(t, g) \tag{2}$$

where t is the tag in question, G_p is the set of all games the user with profile p has played for more than 2 hours. Finally, when the tag ratings have been calculated, for every tag that exists in any of the games user has played, the ratings are scaled between 0 and 1 so that the highest tag rating is always 1. After this we take the top 30 tags per profile and the rest are pruned.

With our method, the tags that have higher strengths and appear more frequently in the user's games get higher tag ratings. As some tags are more frequent and less informative than others, IDF balances the tag ratings between common and uncommon tags.

Our method is limited in such a way, that we only know which tags the user likes, and have no way of telling which tags they dislike. Therefore, a low tag rating does not mean that the user dislikes a tag, but instead the tag is just not as preferable as other tags that have higher tag ratings.

4.2 Generating Single-user Recommendations

After constructing the user profiles, recommendations are generated for each of these profiles separately. For each user profile, every game in the dataset is given a rating, called the game rating. This game rating is determined by the tag preferences of the profile, described by the tag ratings, and the tags the game has.

The rating algorithm works in such a way, that it increases game ratings for games that have multiple preferred tags and penalizes games for tags that the user has no preference for. Overall, the system aims to give equally good ratings for games that have few tags, all of which the profile has high tag ratings for, and games that have a large number of tags but with only moderate tag ratings.

The game rating for a single game is calculated by comparing the tags the game has, to the user profile. The general idea is that the more tags the game contains that the user likes, the better the game rating. If there are many tags but only a few of them are preferred by the user, give a small penalty. This is to avoid only recommending games that have a lot of tags. The game rating for a game g is calculated as:

$$game\_rating(g,p) = \frac{\sum_{t \in T_g} strength(t,g) * tag\_rating(t,p)}{\sqrt[3]{|T_g|}} \quad (3)$$

where g is the game in question and T_g is the set of tags the game g has.

With this formula, games that have multiple tags the user likes, get higher game ratings, but the number of tags the game has also reduces the game rating. This means that games that have a large number of tags might not get better game ratings than games with only few tags. Calculating the game ratings without taking the number of tags into account would have resulted in games with large number of tags dominating the recommendations. On the other hand, also dividing by the number of tags without taking the cube root, in other words taking the average, would result in games with only few tags dominating.

4.3 Generating Group Recommendations

Individual recommendation lists are aggregated into a single group recommendation list using two alternative methods: the *Borda count* and the *least misery* method. The main idea of both methods is to take the single user recommendations of every group member and order the recommendations in such a way that it takes every group member into consideration. For this purpose, each item is given a *group score*, which is then used to rank the items to form the aggregated recommendations. In our case, users are the generated user profiles and items are the games.

Borda count is commonly used in political elections, but is also applicable to recommendations aggregation. It gives somewhat balanced results overall, and in our case games that have high ratings for every user profile get also high group scores, while games that only some of the users like get moderate group scores. Games that none of the group members like get very low group scores. With Borda count, each item is given a score depending on its position in the ranking. The last item in the ranking gets a score of 1 and the first item gets a score of n, where n is the total number of items in the ranking. The group score of an item is calculated by summing up the individual user scores for that particular item:

$$score\_borda\_count(g,i) = \sum_{u \in g} score(u,i) \quad (4)$$

where u is a user in the group g and $score(u,i)$ is the score for the item i, calculated from the ranking.

Least misery is an aggregation method that tries to sort the recommendations in such a way that everyone in the group is satisfied, in other words, causes the least misery among the group. To achieve this, the group score for an item i is the minimum rating for that item from the individual recommendations, as described in: $score\_least\_misery(g, i) = \min_{u \in g}\{score(u, i)\}$.

The items are sorted by their group score to form the final aggregated recommendations.

4.4 Diversifying Group Recommendations

Our goal is to provide diverse results for the users, because this way the chance of finding something interesting is higher. To achieve diverse results, we utilize a k-means clustering-based method for creating clusters of the recommended games. This way, we group similar games and separate dissimilar ones, and by showing results from every one of these clusters evenly, the final result set will be diverse. Because clustering is quite an expensive operation, it is only performed for some of the top recommendations that are generated with the previously described recommendation aggregation. This way, only the games that have good ratings are considered, and there is no need to check later if the clusters contain good recommendations. In our experiments, we used top-500 game recommendations for clustering.

The clustering algorithm works as follows:

1. Get the top-500 recommendations generated with Borda, and prune everything else. These games are hence referred simply as all games.
2. Pick the top-1 recommendation and set it as the first cluster center. This way, we always get a cluster that initially centers on the best game.
3. Pick the most dissimilar game to all the current cluster centers from the top-100 recommendations and make it a new cluster center. This step helps to ensure that the clusters really are dissimilar. This step is repeated until we have 5 cluster centers.
4. Assign every game to their nearest cluster. Distance to every cluster center is measured with the Manhattan distance between their tag strengths.
5. Calculate new cluster centers for each cluster by calculating the averages of tag strengths of each game in that cluster.
6. Steps 4 and 5 are repeated until no more changes happen in the clusters or the maximum number of iterations is reached. 50 iterations is the limit used in our experiments.

5 Usability Evaluation

To evaluate the effectiveness of GameRecs, we implemented it as a web application and performed usability experiments with a number of real users[1]. Specifically, we evaluated the quality of profile generation (Section 5.1), the quality of

[1] Our application implementation and the data used in the experiments are publicly available at https://github.com/Nikkilae/group-game-recommender-test

Table 1: Evaluation of generated user profiles

	u1	u2	u3	u4	u5	u6	u7	u8	u9	u10	u11	u12
PP	93.3%	66.7%	100.0%	93.3%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	80.0%	93.3%
PHPP	40.0%	13.3%	66.7%	80.0%	46.7%	86.7%	53.3%	86.7%	46.7%	80.0%	40.0%	46.7%
PQ	7	4	8	8	7	8	9	8	9	9	7	7

	u13	u14	u15	u16	u17	u18	u19	u20	u21	u22	u23	u24
PP	100.0%	93.3%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	80.0%	93.3%	100.0%	86.7%
PHPP	40.0%	60.0%	73.3%	73.3%	40.0%	26.7%	73.3%	66.7%	46.7%	26.7%	53.3%	46.7%
PQ	7	9	9	7	7	9	8	7	7	8	9	7

	PP	PHPP	PQ
average	95.0%	54.7%	7.7

single user recommendations (Section 5.2), and the quality of group recommendations (Section 5.3).

5.1 Quality of Profile Generation

We asked users to evaluate the precision and quality of the profiles generated by GameRecs. We use three measures to evaluate the quality of a generated profile: Profile Precision, Profile Highly Preferred Precision and Profile Quality.

Users were asked to evaluate the quality of the 30 tags appearing in their profile. For characterizing the quality of the tags, users were asked to rate each of the tags with an interest score in the range [1, 5]. According to these ratings, we calculate **Profile Precision** $PP(u,k)$ for user u as follows:

$$PP(u,k) = \frac{relevant\_tags(u,k)}{k} 100\% \tag{5}$$

where $relevant\_tags(u,k)$ is the number of tags rated 2 or higher by the user u in the top k tags of their generated profile. In our experiments, we used $k = 15$. Furthermore, we reported the number of tags that were rated highly (interest score $>= 4$), and calculated the **Profile Highly Preferred Precision** $PHPP(u,k)$:

$$PHPP(u,k) = \frac{highly\_preferred\_tags(u,k)}{k} 100\% \tag{6}$$

where $highly\_preferred\_tags(u,k)$ is the number of tags the user u rated highly (interest score $>= 4$) in the top k tags of their generated profile. Finally, users were asked to provide an overall **Profile Quality** $PQ(u)$ in the range [1, 10] to indicate their degree of satisfaction of the overall result set including all 15 tags. A high number indicates an accurate representation of the user's taste.

The general impression is that having user profiles generated automatically makes it easier for someone to understand the main idea behind the system, since tags in the profiles act as examples of user preferences. As seen in Table 1, the PP is generally high, 95.0% on average, and in most cases even 100%. The PHPP values are varied, but on average more than half of the tags are highly preferred. Although a high PP seems to lead to a high PQ, their respective values differ from user to user. In conclusion, the profile generation seems quite effective.

5.2 Quality of Single User Recommendations

In addition, to evaluate group recommendations, we study the effectiveness of the recommender system for single users. First, we asked users to count the number of games of the top 20 recommendations that they deemed relevant or interesting. Second, we asked for a general recommendations quality rating between 1 and 10. Third, we asked for a rating of diversity among the recommended games between 1 and 10. We asked for these three ratings twice for different sets of recommendations: with clustering and without clustering. We use three different measures for evaluating the quality of single user recommendations: Recommendations Precision, Recommendations Quality and Recommendations Diversity.

Recommendations Precision $RP(u,k)$ for a user u is calculated as:

$$RP(u,k) = \frac{relevant\_games(u,k)}{k} 100\% \tag{7}$$

where $relevant\_games(u,k)$ is the number of games deemed relevant by the user u in the top k (20) recommendations. **Recommendations Quality** $RQ(u)$ represents the general satisfaction on the generated recommendations given by the user u as a number between 1 and 10. Finally, **Recommendations Diversity** $RD(u)$ represents the general recommendation diversity evaluated by the user u as a number between 1 and 10. As seen in Table 2, on average, just over half of the top recommendations were relevant to the user. As expected, applying clustering reduces RP and increases RD. However, the difference that clustering made in RD is not very impressive.

Table 2: Evaluation of single user recommendations

		u1	u2	u3	u4	u5	u6	u7	u8	u9	u10	u11	u12
With clustering	RP	60.0%	45.0%	75.0%	35.0%	50.0%	50.0%	25.0%	65.0%	25.0%	40.0%	90.0%	75.0%
	RQ	8	7	9	4	4	6	4	8	5	8	5	7
	RD	4	2	7	9	8	9	2	4	3	2	3	7
Without clustering	RP	40.0%	65.0%	65.0%	60.0%	90.0%	40.0%	45.0%	85.0%	45.0%	70.0%	90.0%	50.0%
	RQ	6	8	7	8	9	5	7	8	8	9	4	6
	RD	7	1	8	5	8	3	1	2	6	4	5	6

		u13	u14	u15	u16	u17	u18	u19	u20	u21	u22	u23	u24
With clustering	RP	55.0%	40.0%	60.0%	35.0%	55.0%	40.0%	75.0%	65.0%	25.0%	30.0%	65.0%	85.0%
	RQ	7	4	10	6	5	6	8	8	7	7	8	8
	RD	4	4	7	4	5	8	5	3	8	8	7	8
Without clustering	RP	70.0%	40.0%	80.0%	75.0%	50.0%	60.0%	80.0%	90.0%	30.0%	35.0%	55.0%	95.0%
	RQ	6	9	10	8	5	9	6	9	7	8	7	9
	RD	4	5	4	4	3	6	8	2	8	8	8	5

	With clustering			Without clustering		
	RP	RQ	RD	RP	RQ	RD
average	52.7%	6.6	5.5	62.7%	7.4	5.0

5.3 Quality of Group Recommendations

To evaluate our main focus, the group recommendations, we performed multiple experiments with groups of two and four members. The evaluation was done in four setups, with and without clustering, and using Borda count and least misery aggregation. For each of our four setups, we asked each member of a group to mark each produced recommendation as either relevant or irrelevant. These markings were then used to calculate different quality measures, which are described in more detail below.

Group Recommendations Precision $GRP(g, k)$ for a group g is calculated with the following formula:

$$GRP(g,k) = \frac{relevant\_games\_to\_all(g,k)}{k} 100\% \tag{8}$$

where $relevant\_games\_to\_all(g, k)$ is the number of games in the top k recommendations that were relevant to everyone in the group individually. This means that a recommendation has to be relevant to every group member to be considered as relevant recommendation for the whole group. We used a k of 20 in our experiment.

Partial Group Recommendations Precision is similar to Group Recommendation Precision, but this time the recommendations have to be relevant to only part of the group members to be considered as relevant to the whole group. Partial Group Recommendation Precision $PGRP(g, k)$ for a group g is calculated with the following formula:

$$PGRP(g,k) = \frac{relevant\_games\_to\_half(g,k)}{k} 100\% \tag{9}$$

where $relevant\_games\_to\_half(g, k)$ is the number of games relevant to at least half of the group members from group g from the top k recommendations.

To explore a potential correlation between group similarity and recommendations precision, we measure a group's similarity with Jaccard similarity of tags appearing in group members' profiles. More precisely, it's the relation of the number of common tags shared by the group members' profiles to the number of unique tags that appear in any of the group members' profiles.

The results of group recommendations evaluation can be seen in Table 3. Borda count seems to give slightly better results in both cases. Overall, similarly to single user recommendations, the precision is higher without clustering. Smaller group size seems to lead to higher similarity and better precision.

6 Summary

In this paper, we focus on group recommendations for video games. We propose generating diverse game recommendations for groups of people to play together. We work on open data gathered from Steam, a popular digital distribution platform, and we are capable of producing user profiles based on Steam's data, as

Table 3: Evaluation of group recommendations

	With clustering				Without clustering					
	Borda Count		Least Misery		Borda Count		Least Misery			
	GRP	PGRP	GRP	PGRP	GRP	PGRP	GRP	PGRP	Size	Similarity
[u1,u2,u3,u4]	5.0%	40.0%	5.0%	40.0%	0.0%	40.0%	5.0%	35.0%	4	5.8%
[u5,u6,u7,u8]	15.0%	55.0%	10.0%	45.0%	45.0%	85.0%	30.0%	85.0%	4	17.2%
[u9,u10,u11,u20]	15.0%	75.0%	10.0%	60.0%	10.0%	75.0%	15.0%	70.0%	4	1.2%
[u12,u13,u14,u15]	0.0%	15.0%	0.0%	10.0%	0.0%	5.0%	0.0%	10.0%	4	1.2%
[u16,u17]	35.0%		20.0%		30.0%		25.0%		2	57.9%
[u18,u19]	55.0%		55.0%		75.0%		80.0%		2	36.4%
[u21,u22]	20.0%		15.0%		30.0%		35.0%		2	53.8%
[u23,u24]	50.0%		45.0%		55.0%		45.0%		2	30.4%
average	24.4%	46.3%	20.0%	38.7%	30.6%	51.3%	29.4%	50.0%		

well as video game recommendations for those profiles. To generate group recommendations, we exploit lists aggregation methods, and we target at providing recommendations that exhibit some diversity by using a k-means clustering-based approach. For demonstrating the effectiveness of GameRecs, we performed experiments with real users, and evaluated the usability of the method in terms of the user profile generation and the produced video game recommendations, both for single users and for groups.

Acknowledgement

This work has been partially supported by the Virpa D project funded by Business Finland.

References

1. Adomavicius, G., Kwon, Y.: Multi-criteria recommender systems. In: Recommender Systems Handbook, pp. 847–880 (2015)
2. Balabanovic, M., Shoham, Y.: Content-based, collaborative recommendation. Commun. ACM **40**(3), 66–72 (1997)
3. Beel, J., Langer, S., Genzmehr, M., Nürnberger, A.: Persistence in recommender systems: Giving the same recommendations to the same users multiple times. In: TPDL (2013)
4. Desrosiers, C., Karypis, G.: A comprehensive survey of neighborhood-based recommendation methods. In: Recommender Systems Handbook, pp. 107–144 (2011)
5. Eirinaki, M., Abraham, S., Polyzotis, N., Shaikh, N.: Querie: Collaborative database exploration. IEEE Trans. Knowl. Data Eng. **26**(7), 1778–1790 (2014)
6. Ge, X., Chrysanthis, P.K., Pelechrinis, K.: MPG: not so random exploration of a city. In: MDM (2016)
7. Koskela, M., Simola, I., Stefanidis, K.: Open source software recommendations using github. In: TPDL (2018)
8. McAuley, J.J., Leskovec, J.: Hidden factors and hidden topics: understanding rating dimensions with review text. In: RecSys (2013)

9. Ntoutsi, E., Stefanidis, K., Rausch, K., Kriegel, H.: Strength lies in differences: Diversifying friends for recommendations through subspace clustering. In: CIKM (2014)
10. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: The Adaptive Web, Methods and Strategies of Web Personalization (2007)
11. 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)
12. Sifa, R., Bauckhage, C., Drachen, A.: Archetypal game recommender systems. In: Proceedings of the 16th LWA Workshops (2014)
13. 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)
14. Stefanidis, K., Ntoutsi, E., Kondylakis, H., Velegrakis, Y.: Social-Based Collaborative Filtering, pp. 1–9. Springer New York, New York, NY (2017)
15. Stratigi, M., Kondylakis, H., Stefanidis, K.: Fairness in group recommendations in the health domain. In: ICDE (2017)
16. Stratigi, M., Kondylakis, H., Stefanidis, K.: Fairgrecs: Fair group recommendations by exploiting personal health information. In: DEXA (2018)
17. Yin, Z., Gupta, M., Weninger, T., Han, J.: LINKREC: a unified framework for link recommendation with user attributes and graph structure. In: WWW (2010)