information

MDPI

Article

Context-Aware Music Recommender Systems for Groups: A Comparative Study

Adrián Valera, Álvaro Lozano Murciego and María N. Moreno-García *

Department of Computer Science and Automation, Science Faculty, University of Salamanca, Plaza de los Caídos s/n, 37008 Salamanca, Spain; adrianvalrom.usal@usal.es (A.V.); loza@usal.es (Á.L.M.)
* Correspondence: mmg@usal.es

Abstract: Nowadays, recommender systems are present in multiple application domains, such as e-commerce, digital libraries, music streaming services, etc. In the music domain, these systems are especially useful, since users often like to listen to new songs and discover new bands. At the same time, group music consumption has proliferated in this domain, not just physically, as in the past, but virtually in rooms or messaging groups created for specific purposes, such as studying, training, or meeting friends. Single-user recommender systems are no longer valid in this situation, and group recommender systems are needed to recommend music to groups of users, taking into account their individual preferences and the context of the group (when listening to music). In this paper, a group recommender system in the music domain is proposed, and an extensive comparative study is conducted, involving different collaborative filtering algorithms and aggregation methods.

Keywords: group recommender systems; music recommendation; context-aware recommender systems; collaborative filtering

Citation: Valera, A.; Lozano Murciego, Á.; Moreno-García, M.N. Context-Aware Music Recommender Systems for Groups: A Comparative Study. *Information* **2021**, *12*, 506. https://doi.org/10.3390/info12120506

Academic Editor: Luis Martínez López

Received: 1 October 2021
Accepted: 5 December 2021
Published: 7 December 2021

1. Introduction

Recommender systems (RS) are present today in multiple application domains, such as e-commerce, tourism, television, etc. [1]. Music and audiovisual content, in general, are domains where there is a high demand for these systems, due to the popularization of portable devices and streaming services [1].

Each application area has its own particularities, and the music area is characterized, among other things, by the rapid consumption of items. Consequently, it is difficult to know the ratings explicitly, since users do not stop listening to express them. For this reason, ratings are generally inferred from user behavior. Moreover, the sparsity problem is more important than in other domains, due to the scarcity of ratings and large availability of items in music catalogs.

In this domain, recommender systems for groups acquire great relevance, since music is often listened in company, either by gathering friends for that purpose or while doing another activity, such as training in a gym or working in a library. In contrast, recommendations to groups are not feasible in many other areas where products are used or consumed individually. In addition to music, the main application domains for group recommender systems (GRS) are movies, television, and travel [2]. However, there are differences between these items and music [3]. For example, a song is usually played many times by the same user, while a movie is generally watched only once. Another difference is the number of items consumed in a particular time frame, which is higher for music than for movies and tourist attractions, among other items. These facts influence both the way implicit feedback is obtained from the user and the number and diversity of recommendations.

Moreover, in the music domain, the context of the user also has a great importance on user preferences [4]. Many environment-related contextual factors, such as time, weather, or place, as well as user-related contextual factors, such as, activity or mood, influence

the choice of the music to play by users. Therefore, the rating that a user would give to a song will be different in each context. These factors do not have as much impact on the recommendation of other products, such as books and other consumer goods. Context-aware recommender systems use any information that allows to characterize the user's situation and refine predictions, so that different ratings can be predicted for the same item and user, depending on his or her contextual state.

It has been found that context is widely addressed in the literature about GRS, especially in the area of music, since it has great relevance in determining the group's situation and establishing recommendations [5].

Most GRSs use the individual predictions generated for each group member and then apply some aggregation strategy to provide the most appropriate recommendations to the group. The aggregation methods used in the recommendation of music for groups are very diverse, including variations proposed by some authors for the most commonly used methods [6]. However, no comparative studies on the suitability of these approaches in different contexts are found in this application domain.

The objective of this paper is to contribute to the state of the art of group recommender systems, where the existing literature is very scarce. The work is focused on context-aware recommender systems, since contextual factors are highly relevant in this type of system. However, despite this proven relevance, there are hardly any studies evaluating the effectiveness of group recommendation methods in different contexts. In order to make up for these shortcomings, this work addresses the current state of the art of context-aware recommender systems for groups in the music domain, by analyzing the main recommendation approaches for groups in this area and performing a comparative study of the behavior of different recommendation methods and aggregation strategies in different contexts.

The main contributions of this work are the following:

- To provide an overview of algorithms used in GRS, group types, aggregation methods, evaluation metrics, and context-aware recommender systems.
- To review the current state of GRS in the field of music. This is an application domain where the literature is poor, despite the fact that it is one of the fields where recommendations to groups acquire great relevance.
- To conduct an extensive comparative study of the performance of the eight main aggregation strategies used with the most important collaborative filtering algorithms in different contexts and for different types of groups.

To the best of our knowledge, there is no study of this type in the literature that analyzes so many factors together. We have only found one similar paper [7], in which some recommendation methods for groups were evaluated, but the groups consisted of only three members who listen to music while playing the same video game (always the same). No other context is studied, and no context-aware recommendations are given. In addition, the songs played are among the top 10 favorites of the group members, and all individuals in the sample are students at the same university.

Our study can be useful to determine which collaborative filtering (CF) methods are most appropriate for GRS and which aggregation strategy to use, depending on different factors, such as the CF method used or the length of the recommendation lists. In addition, conclusions can be drawn about the influence on the results of the introduction of context-aware recommendations and their relationship with the way groups are formed.

The paper is structured as follows. Section 2 briefly reviews the state of the art of group recommender systems and context-aware recommendations. Section 3 presents the proposed study about group recommender systems in the music domain. The results of the study are shown and discussed in Sections 4 and 5, respectively. Finally, Section 6 presents the conclusions obtained.

2. State of the Art

Traditionally, recommender systems have focused on individuals; however, in recent times, with the emergence of mobile applications and the popularization of their use,

social networks, collaborative applications, and group-focused recommender systems have proliferated. A GRS precursor was the MusicFX system [8], which aimed to select music for groups of gym users. In the scope of CF methods, PolyLens [9] was the first system developed to provide recommendations to groups. PolyLens was created as an extension of the popular Movielens movie recommender system. Later, GRSs were proposed in other domains, such as television [10,11], travel [12], or tourist activities [13], among others.

A recommender system for groups [14–18] focuses on scenarios in which recommendation solutions are provided to groups of users who have individual preferences. Although, in most features, they overlap with recommender systems for individuals [1], since group recommendations are usually derived from aggregation of individual recommendations, new problems, and considerations arise.

Depending on how groups are considered, GRSs can involve the aggregation of the outputs of recommender systems for individuals (aggregated predictions) [19,20] or simultaneously consider the preferences of all members, by aggregating them to a group profile, for which recommendations are made (aggregated models) [21]. The first approach is the most commonly applied and the one used in more recent studies.

One of the most important elements of discussion when developing a GRS is the choice of the aggregation strategy. Different aggregation strategies and approaches to group recommendation are discussed in [18]. The evaluation of the group recommendation algorithm is essential to identify its usefulness. Therefore, the different approaches and metrics used to evaluate group recommendation models and their relevance are discussed in the literature.

2.1. Types of Groups

Groups, in the context of GRS, can be classified in different ways [14,17], according to different factors related to its members: types of preferences or the reason why the group has been created.

Considering the preferences of the members of the group, these two categories are established:

1. Homogeneous groups: groups whose members have similar interests.
2. Heterogeneous groups: groups whose members have diverse interests, which can become very disparate.

Depending on how the group is created, they can be classified into three categories:

1. Established groups: formed by individuals who explicitly choose to belong to the group because they have certain particular interests.
2. Occasional groups: formed by people who occasionally perform some activity together. Members may have some particular tastes, with respect to a specific topic or activity.
3. Random groups: groups of people who share the same environment at the same time, for a certain period of time, and whose members may not have much relationship with each other.
4. Automatically identified groups: formed automatically by bringing together individuals with similar preferences, such as groups of article reviewers who are proficient in certain topics.

2.2. Algorithms Used

Both RSs and GRSs can be based on the following approaches:

- Content-based recommender systems: content-based filtering [22] applied to groups is based on the idea of recommending new items, similar to those previously consumed or liked by the user. For example, recommending books to a bookstore customer based on characteristics such as genres read or favorite authors.
- Recommender systems based on collaborative filtering (CF): these are based on user-item interactions, in the form of ratings or other behavior from which implicit ratings

are obtained [2]. CF systems use user-item rating matrices to provide recommendations from other user preferences or on ratings received by items. Continuing with the previous example, instead of focusing on the properties of previously read books, the CF algorithm would take into consideration the book ratings of other users similar to him/her when generating the preference for an individual.

CF can be subcategorized into memory- and model-based methods. The former makes use of the full similarity matrix when generating a recommendation, working directly with the dataset, while the model-based approaches use machine learning methods to create predictive models. Memory-based CF can, again, be divided into user- and item-based, depending on the elements considered to find similarities.

There are also hybrid recommendation methods that combine several approaches aiming at performance improvement.

In addition, for GRSs in particular, the following approaches exist:

- Constraint-based recommendations: this type of recommendations uses constraints proposed by group members that must be met in the proposed recommendations.
- Critique-based recommendations: the user is shown reference items, so that they can provide feedback through an iterative process to improve the recommendation.

Most of the work in the GRS literature uses collaborative filtering methods, including k-NN (k-nearest neighbors) and matrix factorization-based algorithms. In [23], a matrix factorization method is applied to recommend a sequence of music tracks for groups. Chen et al. [24] proposed their own collaborative filtering method for group recommendation, which also considers time dynamics in user preferences. SVD (singular value decomposition) was used in [20] to provide playlists for groups in a room-based environment. Hybrid approaches are also proposed in other studies, as in [25], where a method was proposed to recommend radio stations and music to people traveling in a car.

Regarding the second classification, constraint-based recommendation methods have been the target of some work in the literature [26], as well as the problems arising from the constraints [27]. In [14], the authors discuss ways to use this approach in the GRS context, by combining it with different aggregation strategies. Other studies are focused on the category of critique-based recommendations. In [28], the interactive multi-party critiquing (IMPC) recommendation method for groups is proposed. The proposal is based on the critiquing concept to a conversation between individuals. It was implemented in a mobile phone application for recommending restaurants to groups.

2.3. Aggregation Methods

Either through aggregated predictions or aggregated models, an GRS uses aggregation methods to compose the recommendations for the group.

The aggregation strategies listed in Table 1 were proposed in [6]. They are grouped according to three categories: majority-based (M), strategies that are based on the idea that the best recommendation is found by encouraging those items that are most popular; consensus-based (C), aggregation strategies that try to take into consideration the preferences of all members of the group, seeking a more democratic approach; and borderline (B), aggregation strategies that take into consideration a subset of all available users.

In addition to these strategies, which are the most commonly used, there are other proposals. The music recommender system, proposed in [29], uses negative preferences of group members to build user profiles and not recommend songs similar to those categorized as negative. Baskin and Krishnamurthi [30] proposed a preference aggregation algorithm that finds a high-quality consensus ordering of the items by using variable-neighborhood local search. Playlist recommendations for groups is addressed in [23], where a sequential recommendation technique is proposed to build a tracks' sequence iteratively, while constantly balancing user satisfaction levels.

Table 1. Aggregation functions.

Aggregation Strategy	Description	Function
Additive Utilitarian (ADD) [C]	Sum of the predicted ratings for an item.	$argmax_{(t \in I)}(\sum_{u \in G} eval(u,t))$
Multiplicative (MUL) [C]	Multiplication of the predicted values for an item.	$argmax_{(t \in I)}(\prod_{u \in G} eval(u,t))$
Average (AVG) [C]	Average of the predicted values for an item.	$argmax_{(t \in I)}\left(\frac{\sum_{u \in G} eval(u,t)}{\lvert G \rvert}\right)$
Average without Misery (AVM) [C]	Average of predicted predictions that exceed a certain threshold of relevance.	$argmax_{(t \in I))}\left(\frac{\sum_{u \in G} rating(u,t)}{\lvert G \rvert}\right)$ $(t \in I: \nexists u \in G \mid eval(u,t) \leq threshold)$
Approval Voting (APP) [M]	Number of predicted ratings that exceed a certain threshold of relevance.	$argmax_{(t \in I)}(\lvert \{uG : eval(u,t) \geq threshold\} \rvert)$
Most Pleasure (MPL) [B]	Maximum rating of an item.	$argmax_{(t \in I)}(maxeval(t))$
Least Misery (LMS) [B]	Minimum prediction of an item.	$argmax_{(t \in I)}(mineval(t))$
Majority Voting (MAJ) [B]	Maximum prediction of an item.	$argmax_{(t \in I)}(majorityeval(t))$

2.4. Evaluation Metrics

The metrics used to evaluate recommender systems for groups are often similar to those used in RSs. They can be classified into the following two types: rating prediction and top-n recommendations. In the former, the system predicts ratings for the items not consumed previously by the group. In order to evaluate these systems, different types of prediction errors are computed. In the second type, the system provides an ordered list of relevant items to recommend to the group, and the recommendations in this list are evaluated by means of classification metrics and other rank-based measures.

Metrics focused on top-n recommendations tend to be more reasonable, since a system might predict with low error the rating of the undesirable items but be unable to recommend relevant items that the user would consume. An item is considered relevant to a group when it exceeds a relevance threshold.

Among the most used classification metrics are precision and recall. Precision is the fraction of the number of relevant items recommended, in relation to the total number of items recommended, Equation (1):

$$\text{precision@k(g)} = \frac{|\text{predicted}_\text{k}(\text{g}) \cap \text{relevant}_\text{k}(\text{g})|}{\text{k}} \quad (1)$$

where $\text{predicted}_\text{k}(\text{g})$ is the list of k items recommended to the group g, $\text{relevant}_\text{k}(\text{g})$ represents all items relevant to the group g, and k is the number of items recommended.

Recall, or sensitivity, is the fraction of the number of recommended items, relative to the total number of relevant items.

$$\text{recall@k(g)} = \frac{|\text{predicted}_\text{k}(\text{g}) \cap \text{relevant(g)}|}{|\text{relevant(g)}|} \quad (2)$$

The discounted cumulative gain (DCG) is a rank-based metric, where highly relevant items in lower positions in the list should be penalized by decreasing their relevance logarithmically (3) with the position i.

$$\text{DCG@k(g)} = \sum_{i=1}^{k} \frac{2^{relevance(i,g)} - 1}{\log_2(i+1)} \quad (3)$$

However, this value may require normalization if the number of items recommended in the lists varies. This is done in comparison to the ideal DCG (iDCG) given by, Equation (4), obtaining the normalized DCG (nDCG) using Equation (5).

$$\text{iDCG@k} = \sum_{i=1}^{k} \frac{1}{\log_2(i+1)} \quad (4)$$

$$\text{nDCG@k(g)} = \frac{\text{DCG@k(g)}}{\text{iDCG@k}} \tag{5}$$

Another way to evaluate these metrics is, instead of first aggregating the ratings given by the members of each group and evaluating the list, to calculate the metric for a user, with respect to the recommendation. The metric is first computed for all users in the group, and then the average is calculated as shown in Equation (6).

$$\text{metric@k(g)} = \frac{\sum_{u \epsilon g} \text{metric@k(u)}}{|g|} \tag{6}$$

2.5. Context-Aware Recommender Systems

Contextual factors become very important in recommender systems, since an item may be rated very positively by users in one context but be inappropriate in another. There are three main categories of methods for dealing with contextual information: two-dimensional (2D) methods, based on contextual modeling and hybrid.

2D methods: these are the most commonly used methods, the simplest and the ones that need the fewest requirements, since they work with a two-dimensional space:

$$f_{\text{rating}} : \text{User} \times \text{Item} \rightarrow \text{Rating} \tag{7}$$

These methods perform a filtering of the items, according to the contextual requirements before (pre-filtering) or after (post-filtering) the recommendation is made.

Contextual modeling consists of considering all the contextual information at the time the recommendation is made, incorporating it into the algorithm itself.

$$f_{\text{rating}} : \text{User} \times \text{Item} \times \text{Context} \rightarrow \text{Rating} \tag{8}$$

Hybrid systems combine two or more of the above-mentioned filtering systems. There are several papers in the literature that consider the context in recommendations to groups. However, most of them only take into account the context in which the group receiving the recommendations is located. There are some exceptions to this, as in [31], where a context-aware framework for multimedia recommendations is presented. It obtains information from intelligent environmental interfaces (location, emotions, physical conditions, etc.) and combines it with temporal factors, user social information, and user characteristics. Gillhofer and Schedl [32] analyzed user music listening behavior as a function of context, considering many factors, such as time, location, device, weather, etc. From these contextual factors, they are able to detect mood, artists, and genres. Yang et al. [19] proposed a Bayesian approach to find social relationships between users, and this social context is used to detect their favorite categories. In addition, user previous behavior, mood, and so on are included as context.

3. Experimental Study

3.1. Dataset Description

This section details the comparative study, conducted to evaluate the behaviour of different recommendation algorithms and aggregation methods in different groups and contexts. These experiments have been performed employing the surprise python library [33] for the implementation of the CF algorithms and applied to two different datasets, which that we have named:

- LastFM-lk-spotify: this dataset has been obtained by merging the LastFM-1k dataset [34,35] with the song information provided by the Spotify API [36].
- LFM-lb-small-spotify: this dataset has been obtained in a similar way to the previous one, but the dataset published by D. Kowald et al. [37] has been used as the base dataset, which is a subset of the LFM-1b dataset [38].

Both datasets have been obtained using the same process, by merging the user event data with the song information provided by the Spotify API [36]. The records of both datasets contain the following information:

$$\{user\ id, song\ id, rating, \{contextual\ properties\}\}$$

The *user id* and *song id* refer to identifiers of users and songs in each dataset and *rating* to the computed user's score for that song. Although there are no explicit ratings in the original datasets, the existence of listening events, based on timestamps, allows the inference of these through the number of plays of a song. For this purpose, the algorithm proposed by Maciej Pacula in [39] has been used. With this method, explicit ratings can be approximated from implicit ones.

Equation (9) describes the frequency for a user (i) and a song (j) computed from $count(i,j)$, which is the number of plays of the song (j) by the user (i), and $totalCount$, which is the total number of plays of the user (i) for the whole catalogue. Equation (10) describes how to obtain the explicit rating ($r_{i,j}$) of a user (i) for a song (j) in a ranking of size (k), ordered by play frequency, where k' is the rank of the song with most plays. In this way, a score in the interval (0,4) is obtained.

$$freq_{i,j} = \frac{count(i,j)}{totalCount} \tag{9}$$

$$r_{i,j} = 4 \cdot \left(1 - \sum_{k'=1}^{k-1} freq_{k'}(i)\right) \tag{10}$$

The *contextual properties* of the items have been gathered from Spotify. Spotify offers an API [36] through which it is possible to access different data related to users, artists, and songs. In particular, it assigns 13 characteristics to songs: danceability, accousticness, duration, energy, instrumentalness, liveness, loudness, mode, explicitness, speechiness, tempo, and valence. To obtain the final datasets, requests were made for each of the songs in the tuples of both datasets.

These features were used for context inference. Four types of contexts have, therefore, been established: (1) no context, (2) party, (3) fitness, and (4) chill. For the determination of these, thresholds for the contextual properties have been on each defined context:

- No context: Does not consider any property. This context includes all songs, without making any distinction by property.
- Party: A song is considered appropriate for a party environment when the danceability value exceeds the value of 0.65.
- Fitness: A song is considered appropriate for a sport environment when the energy value exceeds 0.90.
- Chill: A song is considered appropriate for relaxation when the energy value is less than 0.10.

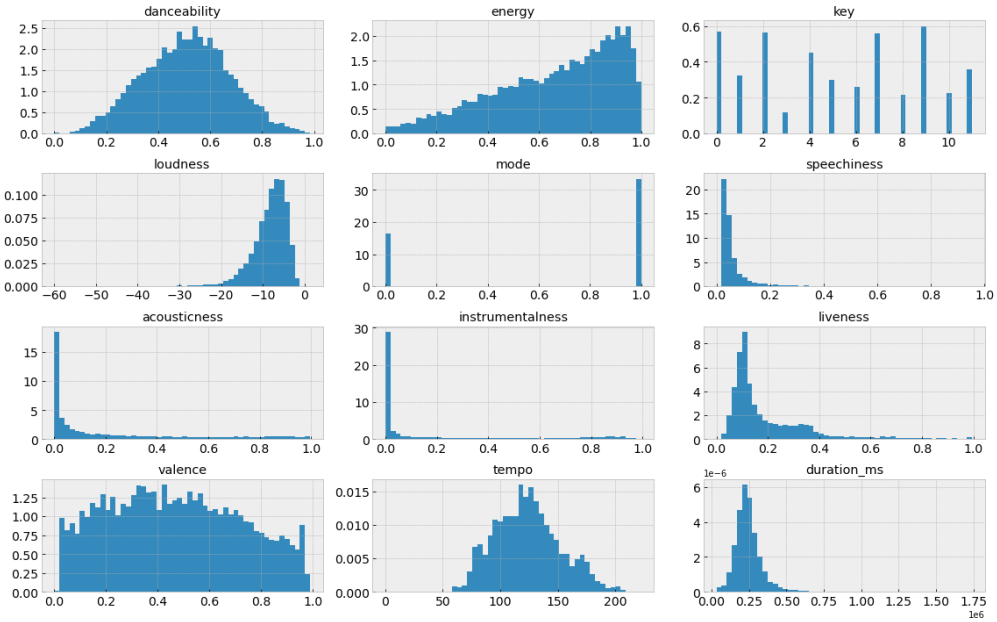
Once the final dataset was obtained, an exploratory data analysis (EDA) was carried out for a first exploration of the data obtained.

An average number of 4.95 ratings per song was detected. Thus, in order to avoid issues in the recommendation, in cases where there are too many songs with few plays (they have been listened to by few users) that are not suitable for collaborative filtering, an additional filtering was carried out on the dataset, establishing a minimum threshold of ratings that a song must have, in order to be used in collaborative filtering. Different tests were carried out and, finally, those songs that had at least 50 ratings, shared by different users, were considered. Once this filtering is done, both datasets were significantly reduced:

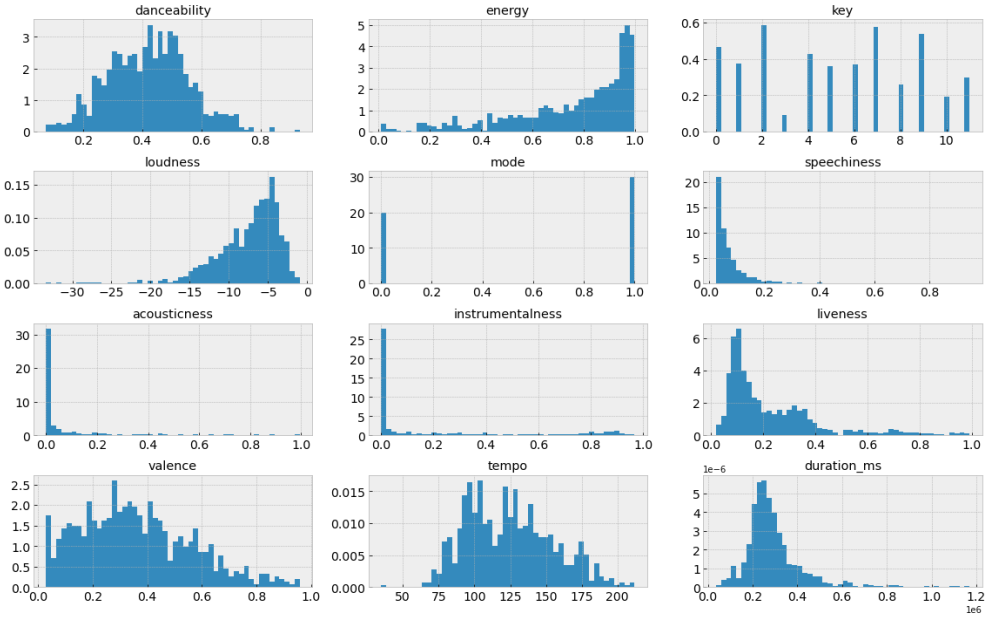
- LastFM-lk-spotify, with a total of 988 users and 9093 songs.
- LFM-lb-small-spotify, with a total of 2713 users and 827 songs.

Figure 1 shows density plots with the distribution of the dataset properties after filtering in both datasets: Figure 1a LastFM-lk-spotify and Figure 1b LFM-lb-small-spotify.

The distribution of song features is similar in both datasets, with slight differences in the danceability valence and tempo features.



(**a**)



(**b**)

Figure 1. Value distribution of song properties in both datasets (**a**) LastFM-lk-spotify and (**b**) LFM-lb-small-spotify.

3.2. Description of the Experiments

Once the datasets were collected, the experiments to perform the comparative study were planned. The aim is to analyse which are the best algorithms, the best aggregation methods, and how the consideration of the context influences the results. Therefore, top-n lists for different groups were obtained from different scenarios, corresponding to all possible combinations of recommendation algorithms, aggregation methods, and defined contexts. The approach chosen for making context-aware recommendations was contextual

pre-filtering, which requires to apply the CF methods after removing from the dataset the examples not relevant for a given context. The lists provided by the algorithms were evaluated, and the results were compared.

For the experiments, 50 groups of 5 members for each of the following possible group types have been considered:

- Random groups: individuals were randomly considered from the total users in the datasets.
- Contextual groups: random individuals were considered after filtering the dataset according to the contexts, and the results obtained were taken from those individuals who had a certain number of relevant songs in that context (fitness, party, and chill), 20 in this case, so that these users can be considered to have a criterion formed in that context.

We have exclusively used the precision and rank evaluation metrics, which, as indicated in the works [5,15], are gaining popularity over the rest, since the error metrics do not provide very useful information when analysing ordered lists. The reason lies in the fact that, in a list of recommendations, and especially in an aggregate recommendation for several users in a group, it is not as important to get accurate predictions as it is to choose those elements that are relevant for the majority and be able to present them to the users as soon as possible.

Once the groups and metrics to be used have been established, recommendations are made for different combinations of the following elements, in order to carry out the experiment:

- 4 contexts: no context, party, fitness, and chill.
- 50 groups of 5 members for each context.
- 4 recommendations algorithms: KNN, NMF, CoClustering, and SVD. All of them included in the surprise library [33].
- 8 aggregation methods: AVG, ADD, MUL, AVM, APP, LMS, MAJ, and MPL.
- 3 evaluation metrics: precision, recall, and NDCG.
- 4 different lengths for top-n lists (@k values): comprised between 5 and 20, hereafter referred to as the @k length of the re-recommendation.

4. Results

The execution of the proposed experiments generated a great volume of results. In order to analyze them, a series of questions have been planned, in order to generate the graphs that could answer them:

1. What are the main results of each algorithm for the relevant metrics (precision, recall, and NDCG)?
2. What are the best aggregation methods, depending on the algorithm, and how does the application of a context affect the values of the metric?
3. What is the best performing algorithm at a fixed @k?

4.1. Main Results of Each Algorithm, in Terms of Precision, Recall, and NDCG

First, we analysed possible biases in the results, since it is important that the metrics obtained in the experiments are not skewed by outlier cases, in order to assure the validity of the results obtained.

In Figure 2, the metric values provided by the different algorithms for an aggregation method are represented by box plots. We can see how, despite the outlier cases, both the mean and median values of the metrics, for each type of group, follow coherent progressions. Figure 2 only shows the case of no context—random groups for MPL (most pleasure) aggregation method, but this behaviour is similar for other contexts and aggregation methods analysed. That is, the SVD algorithm is the one that tends to provide the best results, and the trend we find in the three metrics, as k increases, is roughly similar. Precision and NDCG decrease and recall increases.

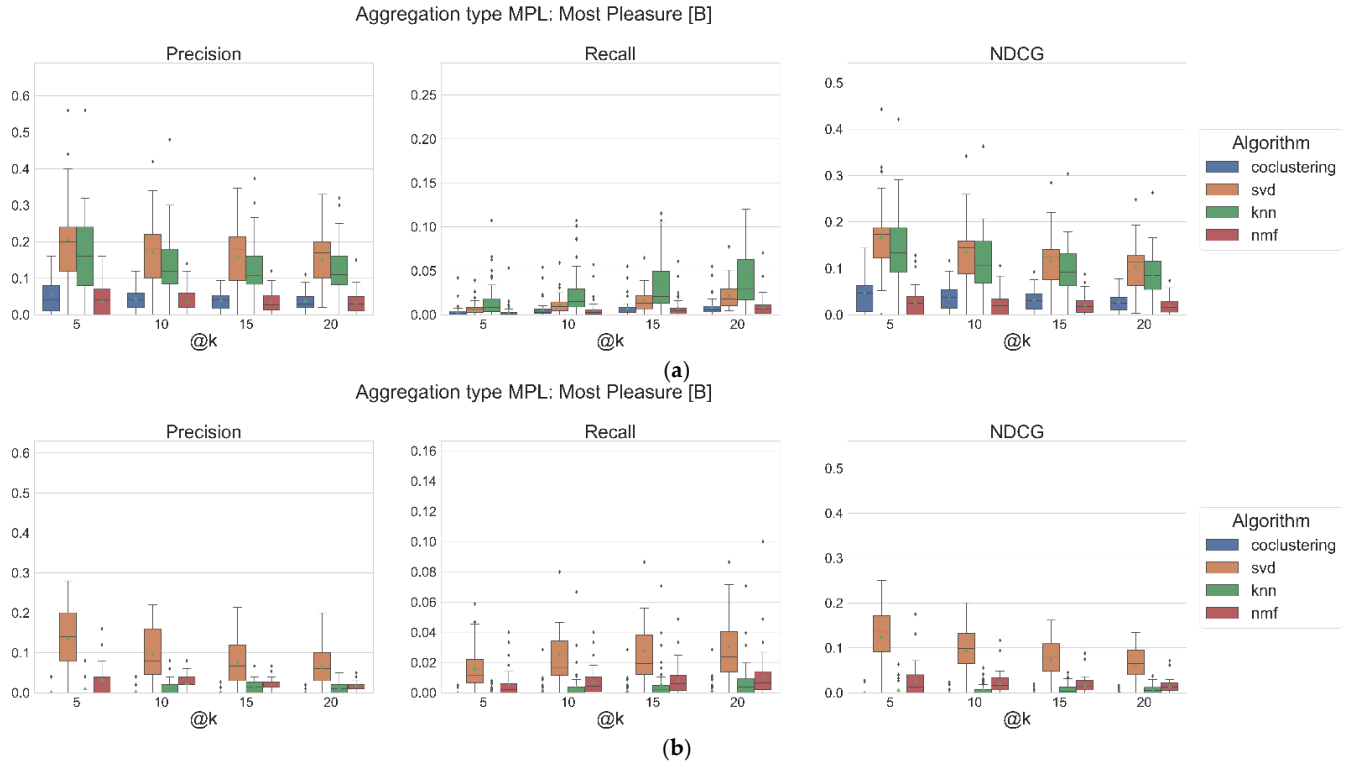


Figure 2. Box plots with the variation in group metrics for applied algorithms at different @k in both datasets: (**a**) LastFM-lk-spotify and (**b**) LFM-lb-small-spotify.

4.2. What Are the Best Aggregation Methods, Depending on the Algorithm, and How Does the Application of a Context Affect the Values of the Metric?

From now on, to analyze the metrics, the median value of the metrics obtained for the types of groups will be considered, since it is less sensitive than the mean to variations, due to atypical cases.

To answer the second question, precision, recall, and NDCG metrics have been obtained for different algorithms, so that it is possible to compare which aggregation methods achieve the highest scores in each case.

We have performed the experiments for all combinations of algorithms (coclustering, SVD, KNN, and NMF), contexts (none, party, fitness, and chill), and group types (random, party, fitness, and chill).

Since there are many possible combinations, we will show (below) the results for the algorithm that has shown the best performance previously, that is, SVD. Figures 3 and 4 show the results for each of the two datasets, respectively, regarding to the metrics for each aggregation method. The combinations shown in these figures are related to chill context; although, in the other contexts, there is a behavior with a similar trend.

Figure 3a shows the precision recall and NDCG results of the SVD algorithm for the context-group type combination none–random, i.e., without contextual pre-filtering and for groups generated randomly, that is, groups of users with interests not linked to any specific context (party, fitness, or chill), randomly selected from dataset. We see that the best performing aggregation methods are MPL [B] (most pleasure (borderline)) and AVM [C] (average without misery (consensus)).

Figure 3b,c show the same data, as previously described, but this time for the context-group type none–chill and chill–chill combinations.

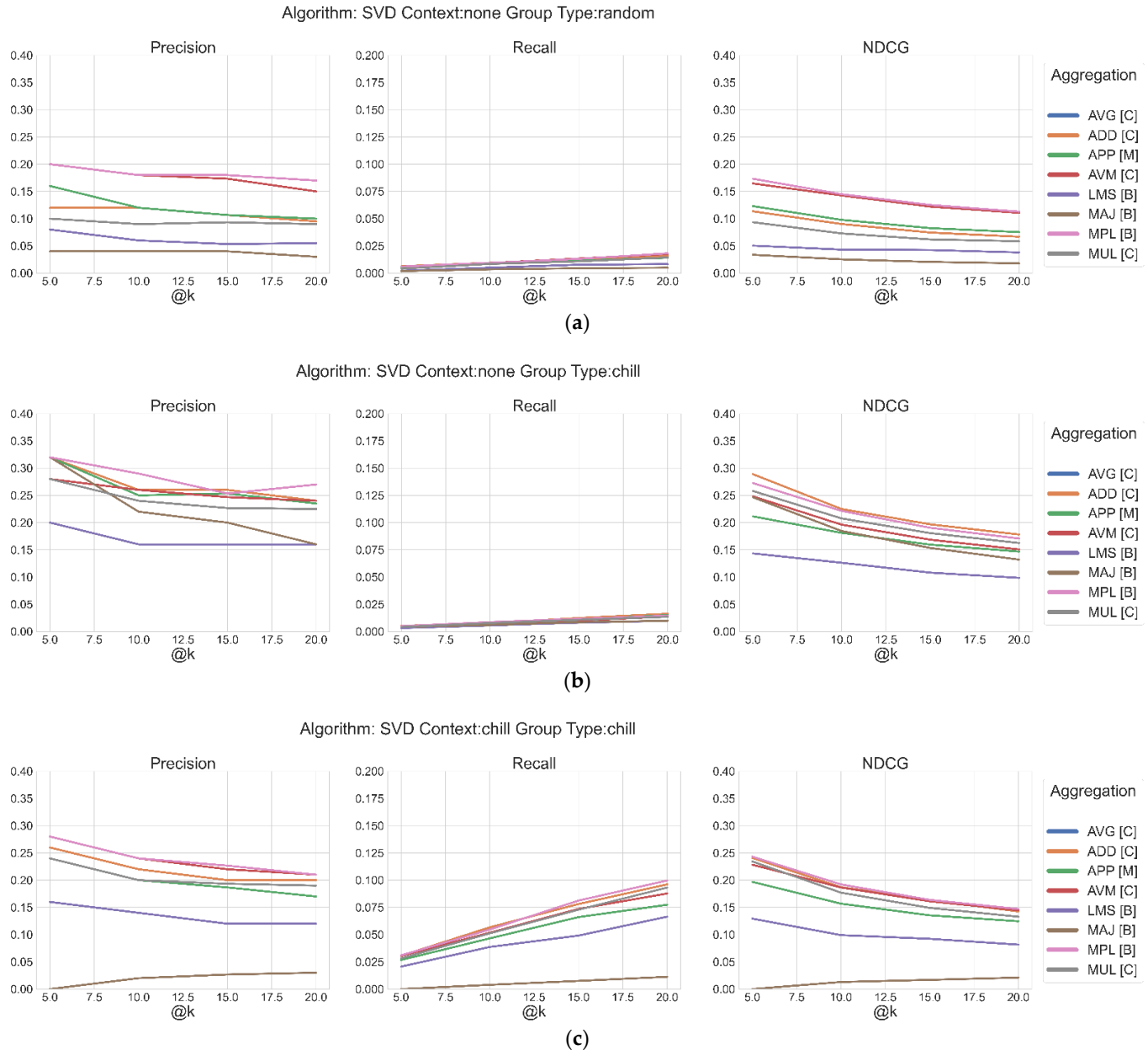


Figure 3. Metric values, depending on aggregation method, with the following combination of context and group type: (**a**) none–random, (**b**) none–chill, and (**c**) chill–chill for LastFM-lk-spotify dataset.

Comparing the three groups of graphs, it is possible to appreciate the improvement of results, when the group type defined is homogeneous, in this case chill, (Figure 3b) and the context and the group type match (Figure 3c), with an increase in recall. This shows a clear influence of the context.

The group of graphs shown in Figure 4a–c corresponds to the same results for the second dataset (LFM-lb-small-spotify). It is possible to see how the aggregation methods mentioned above present the same trend in this case.

For this dataset, the best performing aggregation methods are still MPL and AVM. Additionally, the effect of context can be seen when comparing the three groups of plots.

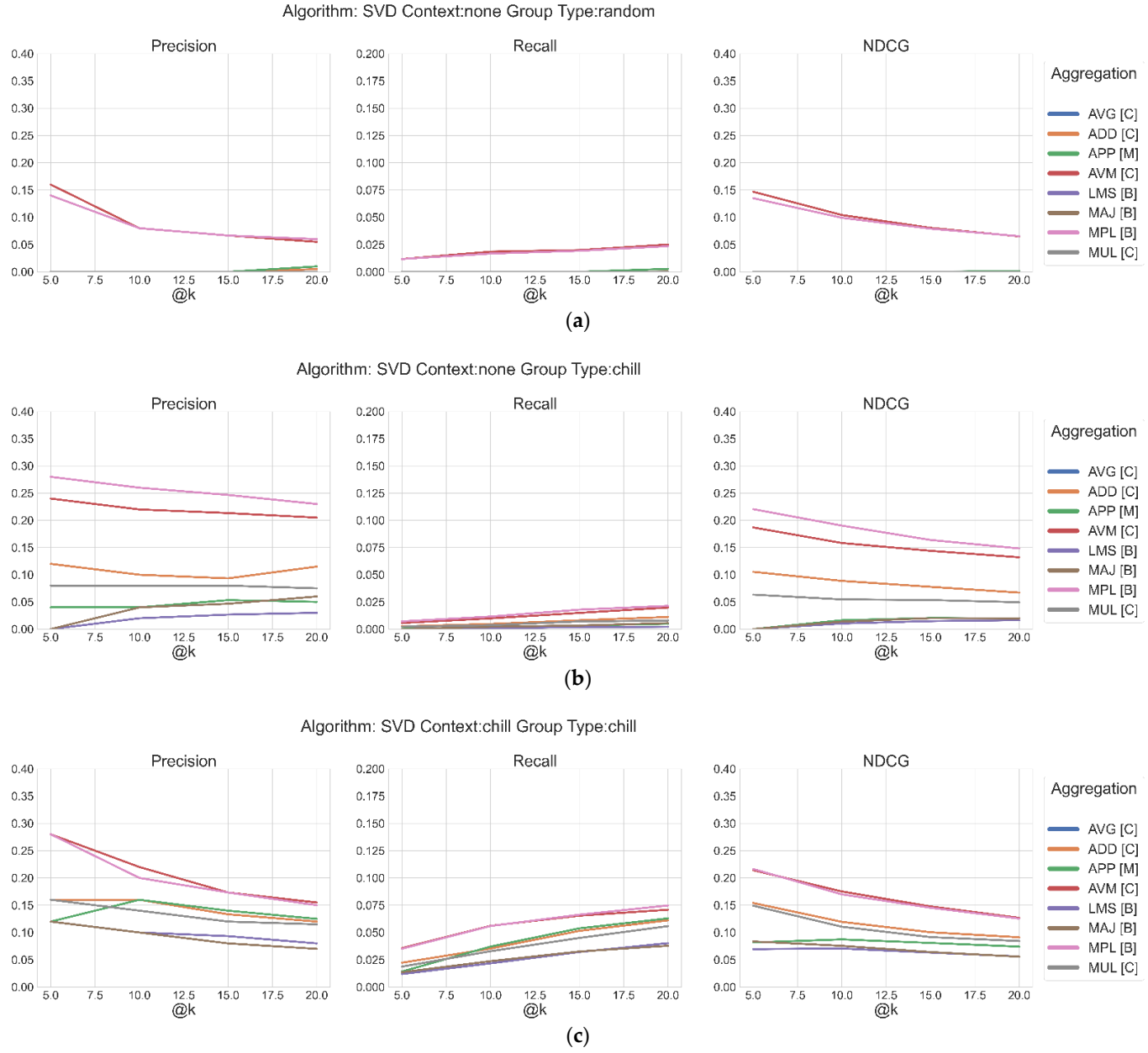


Figure 4. Metric values, depending on aggregation method, with the following combination of context and group type: (**a**) none–random, (**b**) none–chill, and (**c**) chill–chill for LFM-lb-small-spotify.

4.3. What Is the Best Performing Algorithm at a Fixed @k?

The aggregation methods lead to different metric values, depending on the length of the recommended list. For certain aggregations, it is much more complicated to retrieve the relevant items when the list is short (such as LMS with SVD for @5 lists), compared to other aggregations that allow a high precision at the beginning, but that quickly decays when the size of the list increases (such as AVM with SVD algorithm in both datasets).

In addition, we can set the value of @k to make the comparison with a fixed recommendation list length at 20 items (a common amount for the context of Music) and compare the results, depending on whether the recommendation is made for a given context or not (Figures 5 and 6 for the two datasets).

Figure 5. Comparison of metrics, based on aggregation methods, and algorithms for a fixed context in top-n@20 recommendations for LastFM-lk-spotify.

Figure 6. Comparison of metrics, based on aggregation methods and algorithms, for a fixed context in top-n@20 recommendations for LFM-lb-small-spotify.

In these plots, it is possible to observe, at a glance, how some aggregation functions do not perform well with some algorithms (but with others, they do). In addition, when applying pre-filtering for different contexts, each of the aggregation methods behaves in a different way. The NMF algorithm is the one that shows the most invariant and stable results for the different aggregation methods and contexts.

It should be noted that the aggregation functions could be consensus-based [C], majority-based [M], or boundary-based [B]. The aggregation functions APP [M] and MAJ [B] show poor results, in general, for all algorithms. The best aggregation methods are MPL [B] and AVM [C], with the SVD algorithm; however, NMF and SVD combines better with a larger number of aggregation methods. The consensus metrics perform well in many cases, these are: AVG, ADD, MUL, and AVM.

However, this representation is not totally fair because an aggregation method may yield good results for a given depth of top-n lists (such as @20) but perform worse than others at a larger extent (@100).

It was also found that, depending on the aggregation model and the algorithm used, the recommendation either make sense or do not. For example, if the MUL aggregation and the KNN algorithm are used, the precision at @5 is close to 0 and less than 0.05 in the first third of the list. The coclustering algorithm shows quite poor metrics, compared to other algorithms, which has also been seen in the graphs of the previous questions.

Results are highly dependent on context, algorithms, aggregation methods, and lengths of the top-n lists evaluated. Therefore, we can conclude that, depending on the particular case to be implemented, it may be interesting to consider one metric or another, since they behave differently in different situations. However, the method providing the best results, with and without context and for most aggregation strategies, was SVD.

5. Discussion

The results shown in the previous section have helped to answer the research questions posed previously. Firstly, it has been shown that the trends in the metrics are similar for the different experiments carried out, and the best performing algorithm is SVD.

The results also showed that group recommendations are significantly improved for all aggregation methods when groups formed for a given context receive context-aware recommendations, compared to those provided to randomly created groups.

Additionally, the behavior of both CF algorithms and aggregation strategies varies considerably with the length of the top-N lists. More specifically, the good behavior of SVD was detected, regardless the context, type of group, and aggregation method. Additionaly, KNN works well for a given context but is outperformed by NMF and SVD when the list to be recommended is short. It has also been found that the NMF algorithm gives acceptable results for both precision and NDCG, with a higher number of aggregation functions than KNN.

Moreover, it has been observed that some aggregation functions do not work with some algorithms, with NMF and SVD algorithms showing the most stable results for different aggregation methods and contexts. In addition, MPL and AVM are the best performing aggregation strategies with almost all CF methods, especially with the SVD algorithm; although, the precision worsens when top-n@k recommendations are made for a small values of k.

Regarding the evaluation metrics, if we consider a real scenario, it is interesting to look for those with high NDCG when the objective is the generation of playlists, generally short, in which the songs must have a correct sorting and relevant elements at the beginning. On the contrary, when the objective is to generate music recommendations for a bar, it is better to focus on high precision and recall, which translate into a large number of relevant recommendations in long top-n lists, since the music will be playing for a longer time and the items are not consumed so immediately.

6. Conclusions

In this work, the main aggregation methods for group recommendations and evaluation metrics have been reviewed. Additionally, an exhaustive experimental study in the music domain has been conducted to validate the different aggregation strategies, when used in combination with the most popular CF methods for both context-free and -aware recommendations, as well as with randomly formed groups and groups formed for a given

context. The results obtained with several of the CF algorithms and aggregation methods tested were good. However, different combinations of algorithms and aggregation methods can be considered, depending on the problem to be solved, since the results are different for different contexts and types of groups. Therefore, it is interesting to use a benchmark, such as that presented in this work, when making recommendations in a real scenario.

Author Contributions: Conceptualization, A.V., Á.L.M. and M.N.M.-G.; methodology, Á.L.M. and M.N.M.-G.; software, A.V.; validation A.V. and Á.L.M.; investigation, A.V., Á.L.M., and M.N.M.-G.; data curation, A.V.; writing—A.V.; writing—review and editing, Á.L.M. and M.N.M.-G.; visualization, A.V. and Á.L.M.; supervision Á.L.M. and M.N.M.-G.; project administration, M.N.M.-G.; funding acquisition, M.N.M.-G. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been supported by the Department of Education of the Junta de Castilla y León, Spain, (ORDEN EDU/667/2019). Grant number: SA064G19.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Information about the datasets is given in the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ricci, F.; Shapira, B.; Rokach, L. *Recommender Systems Handbook*, 2nd ed.; Springer: New York, NY, USA, 2015; pp. 1–1003.
2. Aggarwal, C.C. *Recommender Systems*; Springer: Cham, Switzerland, 2016.
3. Schedl, M.; Zamani, H.; Chen, C.W.; Deldjoo, Y.; Elahi, M. Current challenges and visions in music recommender systems research. *Int. J. Multimed. Inf. Retr.* **2018**, *7*, 95–116. [CrossRef]
4. Schedl, M.; Knees, P.; McFee, B.; Bogdanov, D.; Kaminskas, M. Music Recommender Systems. In *Recommender Systems Handbook*; Springer: New York, NY, USA, 2015; pp. 453–492.
5. Murciego, Á.L.; Jiménez-Bravo, D.M.; Román, A.V.; Santana, J.F.D.P.; Moreno-García, M.N. Context-aware recommender systems in the music domain: A systematic literature review. *Electronics* **2021**, *10*, 1555. [CrossRef]
6. Valera, A.; Murciego, A.L.; Moreno-Garcia, M.N. Group Recommender Systems in the Music Domain: A Systematic Literature Review. In *New Trends in Disruptive Technologies, Tech Ethics and Artificial Intelligence. DiTTEt 2021*; de Paz Santana, J.F., de la Iglesia, D.H., López Rivero, A.J., Eds.; Advances in Intelligent Systems and Computing; Springer: Cham, Switzerland, 2022; Volume 1410, pp. 296–307.
7. Mezei, Z.; Eickhoff, C. Evaluating Music Recommender Systems for Groups. *arXiv* **2017**, arXiv:1707.09790.
8. McCarthy, J.F.; Anagnost, T.D. MusicFX: An arbiter of group preferences for computer supported collaborative workouts. In Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work, Washington, DC, USA, 14–18 November 1998; pp. 363–372.
9. O'Connor, M.; Cosley, D.; Konstan, J.A.; Riedl, J. PolyLens: A Recommender System for Groups of Users. In *ECSCW 2001*; Springer: Dordrecht, The Netherlands, 2001; pp. 199–218.
10. Masthoff, J. Group Modeling: Selecting a Sequence of Television Items to Suit a Group of Viewers. In *Personalized Digital Television*; Springer: Dordrecht, The Netherlands, 2004; pp. 93–141.
11. Yu, Z.; Zhou, X.; Hao, Y.; Gu, J. TV program recommendation for multiple viewers based on user profile merging. *User Model. User-Adapt. Interact.* **2006**, *16*, 63–82. [CrossRef]
12. Smyth, B.; Mcginty, L.; Reilly, J.; Mccarthy, K. Compound Critiques for Conversational Recommender Systems. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI'04), Beijing, China, 20–24 September 2004.
13. Garcia, I.; Sebastia, L.; Onaindia, E.; Guzman, C. A group recommender system for tourist activities. In Proceedings of the International Conference on Electronic Commerce and Web Technologies, Linz, Austria, 1–4 September 2009; Volume 5692, pp. 26–37.
14. Delic, A.; Masthoff, J. Group recommender systems. In *UMAP 2018, Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, Singapore, Singapore, 8–11 July 2018*; ACM: New York, NY, USA, 2018; pp. 377–378.
15. Boratto, L.; Carta, S. State-of-the-art in group recommendation and new approaches for automatic identification of groups. *Stud. Comput. Intell.* **2010**, *324*, 1–20.
16. Dara, S.; Chowdary, C.R.; Kumar, C. A survey on group recommender systems. *J. Intell. Inf. Syst.* **2020**, *54*, 271–295. [CrossRef]
17. Kompan, M.; Bielikova, M. Group recommendations: Survey and perspectives. *Comput. Inform.* **2014**, *33*, 446–476.
18. Jameson, A.; Smyth, B. Recommendation to groups. In *The Adaptive Web*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4321, pp. 596–627.

19. Yang, Q.; Zhan, L.; Han, L.; Zhang, J.; Bi, Z. Recommending more suitable music based on users' real context. In Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing, London, UK, 19–22 August 2019; Volume 268, pp. 124–137.
20. Li, H.W.; Sou, S.I.; Hsieh, H.P. Room-based Playlist Arrangement System using Group Recommendation. In Proceedings of the 2020 International Computer Symposium (ICS), Tainan, Taiwan, 17–19 December 2020; pp. 50–54.
21. De Carolis, B.; Ferilli, S.; Orio, N. Recommending music to groups in fitness classes. Music Technology Meets Philosophy: From Digital Echos to Virtual Ethos. In Proceedings of the 40th International Computer Music Conference Joint with the 11th Sound and Music Computing Conference, Athens, Greece, 14–20 September 2014; pp. 1759–1765.
22. Deldjoo, Y.; Schedl, M.; Knees, P. Content-driven Music Recommendation: Evolution, State of the Art, and Challenges. *arXiv* **2021**, arXiv:2107.11803.
23. Piliponyte, A.; Ricci, F.; Koschwitz, J. Sequential music recommendations for groups by balancing user satisfaction. In Proceedings of the UMAP Workshops, Rome, Italy, 10–14 June 2013; Volume 997.
24. Chen, J.; Liu, Y.; Li, D. Dynamic group recommendation with modified collaborative filtering and temporal factor. *Int. Arab J. Inf. Technol.* **2016**, *13*, 294–301.
25. Liu, N.H. Design of an intelligent car radio and music player system. *Multimed. Tools Appl.* **2014**, *72*, 1341–1361. [CrossRef]
26. Felfernig, A.; Burke, R. Constraint-based recommender systems: Technologies and research issues. In Proceedings of the 10th International Conference on Electronic Commerce, Innsbruck, Austria, 19–22 August 2008; ACM: New York, NY, USA, 2008.
27. Felfernig, A.; Schubert, M. A Diagnosis Algorithm for Inconsistent Constraint Sets. In Proceedings of the 21st International Workshop on the Principles of Diagnosis, Held Jointly with the Annual Conference of the PHM Society 2010, Portland, OR, USA, 13–16 October 2010; pp. 1–8.
28. Guzzi, F.; Ricci, F.; Burke, R. Interactive multi-party critiquing for group recommendation. In Proceedings of the Fifth ACM Conference on Recommender Systems, Chicago, IL, USA, 23–27 October 2011; ACM: New York, NY, USA, 2011; pp. 265–268.
29. Chao, D.L.; Balthrop, J.; Forrest, S. Adaptive radio: Achieving consensus using negative preferences. In Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work, Sanibel Island, FL, USA, 6–9 November 2005; pp. 120–123.
30. Baskin, J.P.; Krishnamurthi, S. Preference aggregation in group recommender systems for committee decision-making. In Proceedings of the Third ACM Conference on Recommender Systems, New York, NY, USA, 23–25 October 2009; pp. 337–340.
31. Alhamid, M.F.; Rawashdeh, M.; Dong, H.; Hossain, M.A.; Alelaiwi, A.; El Saddik, A. RecAm: A collaborative context-aware framework for multimedia recommendations in an ambient intelligence environment. *Multimed. Syst.* **2016**, *22*, 587–601. [CrossRef]
32. Gillhofer, M.; Schedl, M. Iron maiden while jogging, debussy for dinner? An analysis of music listening behavior in context. In Proceedings of the International Conference on Multimedia Modeling, Sydney, NSW, Australia, 5–7 January 2015; Volume 8936, pp. 380–391.
33. Hug, N. Surprise: A Python library for recommender systems. *J. Open Source Softw.* **2020**, *5*, 2174. [CrossRef]
34. LastFM. LastFM 1k Dataset. Available online: http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html (accessed on 15 October 2021).
35. Celma, Ò. Music recommendation and discovery: The long tail, long fail, and long play in the digital music space. *Music Recomm. Discov.* **2010**, *1*, 1–194.
36. Spotify Spotify for Developers. Available online: https://developer.spotify.com/ (accessed on 15 October 2021).
37. Kowald, D.; Schedl, M.; Lex, E. The unfairness of popularity bias in music recommendation: A reproducibility study. *Adv. Inf. Retr.* **2020**, *12036*, 35–42.
38. Schedl, M. The LFM-1b dataset for music retrieval and recommendation. In Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval, New York, NY, USA, 6–9 June 2016; ACM: New York, NY, USA, 2016; pp. 103–110.
39. Pacula, M. A Matrix Factorization Algorithm for Music Recommendation Using Implicit User Feedback. 2009. Available online: mpacula.com (accessed on 1 October 2021).