Fair Sequential Group Recommendations

Maria Stratigi
Tampere University
Tampere, Finland
maria.stratigi@tuni.fi

Jyrki Nummenmaa
Tampere University
Tampere, Finland
jyrki.nummenmaa@tuni.fi

Evaggelia Pitoura
University of Ioannina
Ioannina, Greece
pitoura@cs.uoi.gr

Kostas Stefanidis
Tampere University
Tampere, Finland
konstantinos.stefanidis@tuni.fi

ABSTRACT

Recommender systems have been incorporated in our everyday life; from music to health recommendations, recommender systems have enhanced the users' experience. At the same time, with the expansion of social media, it is now easier than ever to form groups of people. As such, group recommenders have become more popular. Often, we consider the interaction between a group and the recommender system as a stand-alone process; the group requests some suggestions from the system and the system answers without any considerations to past interactions. A more realistic scenario is for a system to require access to history logs, and take them into account when recommending items for a group of users. Not only what items the system previously had recommended, but also, how well were these items received by the members of the group. In this work, we propose a sequential group recommender, which is aware of the past interactions of the group with the system. We introduce the notion of *satisfaction* that describes how relevant are the recommended items to each member of the group. We utilize satisfaction in a novel aggregation method that achieves to make our model fair for all members of the group. We show with experimental results that the typical group recommendation approaches are substandard to our proposed method.

KEYWORDS

Recommender systems, sequential recommendations, group recommendations, fair recommendations.

1 INTRODUCTION

Recommendations have been integrated into many of the services available to users in recent times. From listening to music to health information, recommender systems are employed to make the user experience better and smoother. In most cases, recommender systems provide users with a list of data items that are most relevant to them. Two main methods appear to produce such lists of items. The content-based method [24] and the collaborative filtering method [29]. In the content-based case, the system recommends to the user items that are similar to other items that the user has already consumed in the past. This requires previous knowledge about the items which is often hard to obtain. In contrast, collaborative filtering (CF) requires data regarding user preferences for specific items. The main idea behind a CF recommender system is the following: given a target user, the system finds similar enough users to him/her, often called *peers*. Then the items that the peers have shown a preference for (mostly in the form of ratings, but often other forms of feedback are used, such as textual review and binary format of like/dislike) are examined and used as input into a relevance function, which produces the list of relevant items to the target user. Collaborative filtering is a very powerful tool that enables recommender systems to provide far more accurate and specialized recommendations [3].

With the expansion of social media, another form of recommendations has emerged; namely the group recommendations [1, 21, 22]. Instead of a single user requesting recommendations from the system, a group can make a query as well. A standard example of group recommendations is the following: a group of friends wants to see a movie. Each friend has his/her own likes and dislikes. The system needs to properly balance them, and offer to the group a list of items that has a degree of relevance to each member. Recently, there is some research on the selection process of the items. One approach is to create a pseudo user by combing the data of each group member, and then apply a standard recommendation method. The second and most used approach is to apply a recommendation method to each member individually, and then aggregate the separate lists into one for the group. The aggregation phase of the approach is the object of much research done on group recommendations.

There are many different criteria one can take into account during the aggregation stage. One such criterion is *fairness* [19, 30–32]. A basic definition of fairness is to recommend the item that has the best relevance to the group. Instinctively, one way to produce such an item is to calculate the average score across all the group members' preference scores for that item. In such a way, all members of the group are considered equals. However, this has a big drawback. Consider, for instance, the following scenario with a group of three users. Two of them are quite similar to each other, while the third is not. By using the average method the opinion of the last user is lost. An alternative approach is to use the minimum function rather than the average. This way, the user with the minimum preference

SAC'20, March 30-April 3, 2020, Brno, Czech Republic
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-6866-7/20/03…$15.00
https://doi.org/xx.xxx/xxx_x

score will act as a veto to the rest of the group. The least misery approach, as it is called, has a drawback as well. It only takes the opinion of one group member under consideration, while ignoring the others. Subsequently, the system in most cases, recommends an item that is acceptable for all members, but it will be an item with a somehow *low* preference score. The need to combine the equality of the average method and the inclusivity of the least misery method is the driving force behind our work in this paper.

The previous approaches for computing recommendations have a hidden element to them. We imply that each time a group is using the system is distinct from the previous ones and that the system has only one state without keeping a history log. But this is not a realistic scenario. A user, or in our case a group of users, interacts with the system multiple times. So, the system should recommend different items at each iteration, while retaining knowledge from past interactions, so as to keep the output diverse. That is, we do not want to always recommend the same item or the same set of items. We want a system that has a memory and can adjust its recommendations accordingly. Continuing with a group movie example, where the same group of friends requests a movie recommendation each week, both the average and least misery approaches fail through all iterations of the system. In the case of average, the outlier user is never satisfied, while in the case of least misery, the system recommends movies that are not highly relevant to anyone in the group.

In this work, we address the problem of unfairness that is generated by these methods when applied to sequential group recommendations. A simple working scenario is as follows: we have a group of friends that meets in regular times to watch a movie. We want to recommend each time a new movie for the group. Figure 1 gives an experimental example of such a scenario. We take a group of 5 members that ask the system for recommendations 5 different times. Each time the system reports to the group 10 items. In Figure 1-left, we have formed the group list, by using the average aggregation method, while in Figure 1-right, the least misery one. In this example, we count the degree of *satisfaction* for each member, which is calculated by measuring how relevant are the group list's items, over the best items for each member[1]. In both scenarios, User 4 has a very low satisfaction score (for least misery is always 0), which implies that almost none of the reported items are of interest to him/her. It is evident that the recommender system is unfair to him/her, and that unfairness continues throughout the iterations. Ideally, each new group recommendation should take into account what has happened in the past. Not only what movies have already been recommended, but at which degree each member of the group was satisfied with that recommendation.

The notion of multiple iterations of recommendations allows us to make a conjecture. If a user is not satisfied with a particular round of recommendations, then he/she was either satisfied in a previous or will be satisfied in the next round. We introduce the notion of *satisfaction* to estimate the degree of satisfaction in the recommendations for each group member, after each iteration of the system. We apply this satisfaction score during the aggregation phase of the group recommendation method as a weighting function on the individual preference scores of the group members. This way, the users that were not satisfied in the previous round will have more weight in the next iteration. Additionally, a member is not continuously biased against (as may be the case of simple average), since the calculation of the satisfaction scores is done dynamically at each iteration. Finally, we take into account the opinions of the entire group (something that least misery was lacking).

The contributions of our work are the following:

- We introduce the notion of sequential group recommendations. We propose that when adding the dimension of multiple iterations to typical group recommendation approaches, such as the average and least misery aggregation methods, the results do not ensure user satisfaction for all group members.
- We propose the concept of *satisfaction*. Each member of the group has a degree of satisfaction for the items recommended at each iteration, as well as an overall satisfaction, gained by all the previous iterations of the group.
- We propose a sequential group recommendation model that takes into account the previous interactions of the group with the system and alters the influence that a group member has on the formation of the group recommendation list.
- We experimentally show that our proposed model is superior to the standard group recommendation approaches.

The rest of the paper is structured as follows: Section 2 describes the related work. Section 3 presents basic concepts on recommender systems, and Section 4 introduces our approach for sequential group recommendations, targeting at ensuring fair results for all group members. Section 5 presents our experimental setup, and Section 6 presents our evaluation results. Finally, Section 7 concludes the paper with a summary of our contributions and directions for future work.

2 RELATED WORK

A recommender system aims to provide to a user, items that are relevant to him/her, by exploiting already available user information – profile, preferences, etc. One of the most used approaches for producing recommendations is the collaborative filtering approach. With more than a decade of research in the area, there are many different solutions to the problem [3]. Most of them can be divided into two main categories: memory-based and model-based algorithms. Memory-based algorithms [15, 28], employ a user-ratings matrix that contains the ratings each user has given to items. They utilize this matrix to find similar users to a target user, by applying a similarity function. The final prediction is made by examining the ratings of similar users, or as they often called *neighbors or peers*. Model-based algorithms [7], first construct a model to represent the behavior of the users and, therefore, to predict their ratings. In this work, we utilize memory-based collaborative filtering.

2.1 Group Recommendations

Group recommendation is another field with a significant research background. There are two main approaches to group recommendation: virtual user and recommendation aggregation [13]. In the former approach, we combine the profiles and ratings of each group member to form a virtual user so that a standard recommendation approach can be applied. In the latter approach, we apply a standard

[1]For more details see Section 4 – Equation 1.

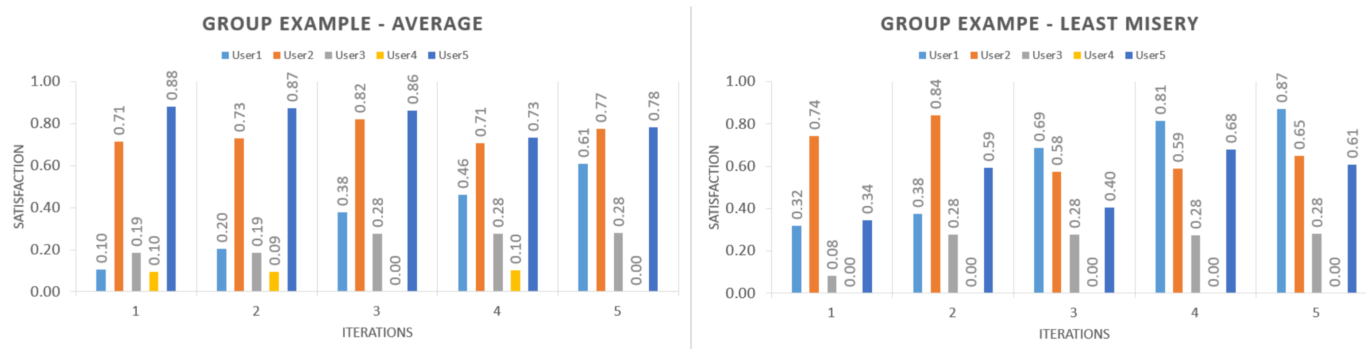


Figure 1: Example for a group with 5 members, and a group recommendation list with 10 items. We consider 5 iterations. We utilize the average (left) and least misery (right) aggregation methods.

recommendation algorithm to each group member individually and aggregate their lists into one. In this work, we follow the latter approach, since it is more flexible [23] and offers opportunities for improvements in terms of effectiveness.

During the aggregation phase of this approach, many criteria can be taken into account. [36] proposes a group recommendation model that takes into account the influence that each member has on the final choice for the group. They state that a member has more influence on the group if he/she is more knowledgeable about the items that are recommended. In our work, we propose that, if there is a group member that is more dissatisfied than the rest, then that member will have more influence in the group decision. [4] learns the aggregation strategy from data, which is based on the recent developments of attention network and neural collaborative filtering (NCF), while we dynamically change our proposed aggregation method based on the satisfaction of the group members. While we focus on a relatively small group of friends (5 in our experiments), [25] offers a novel approach to producing recommendations for a large group of people, by dividing the big group into different interest subgroups. For each subgroup, they find a potential candidate set of media-user pairs and finally aggregate the CF produced recommendation lists for each pair. [16] proposes a two-phase group recommender that, similar to our work, tries to satisfy all the group members. In the first phase, they try to satisfy the whole group, while in the second they try to satisfy the members individually, by filtering out items that are irrelevant to each member. We incorporate these two phases into our proposed aggregation method.

2.2 Fairness in Group Recommendations

There are many different approaches to achieving fairness in group recommendations. One approach is to calculate fairness based on the game and voting theory. [5] solves conflicts of interest between members of heterogeneous random groups by utilizing the Non-Cooperative Game Theory [34]. [20] assumes that the recommender system has probabilistic knowledge about the distribution of users' ratings, and utilizing the voting theory recommends to the group a "winning" item. Finally, [9] proposes a new group recommendation method by allowing a group member to commend on the choices of the rest of the group. This allows each user to get new recommendations similar to the proposals made by other group members and to communicate the rationale behind their own counterproposals.

Another approach to achieving fairness in group recommendation is presented in [1]. It introduces the *consensus function* that similarly to our work takes into account the opinion of the group as it is given by the average method and the disagreement between users. They define the disagreement of users as either the average of pair-wise relevance differences for the item among group members or a variance disagreement function that computes the mathematical variance of the relevance scores for the item among group members. [30] proposes two definition of fairness: *fairness proportionality* and *envy-freeness*. In the former, the user u considers the list of recommended items fair for him/her, if there are at least m items that the user likes. In the latter, u considers the package fair, if there are at least m items for which the user does not feel envious. [35] presents yet another definition of fairness. They define a utility score for each group member based on the relevance that the recommended items have on them. They model fairness as a proximity of how balanced the utilities of users are when group recommendations are given.

All of the works mentioned above in achieving fairness only consider one instance of group recommendations and do not take into account the sequential group recommendation problem, as we do in this work.

2.3 Sequential Recommendations

Sequential recommendations is a relatively new area of research. In general, there are three categories of sequential recommenders, and they are divided based on how many past user interactions they consider: *Last-N interactions-based recommendations, Session-based recommendations* and *Session-aware recommendations* [26]. In the first approach, only the last N user actions are considered [6, 17, 18]. This is because the system has logged a huge amount of historical data for the user, with many of them be duplicates, which do not offer relevant information to the system. Last-N interactions-based recommendations are typically location-aware recommendations. In session-based recommendations, only the last interaction of the user with the system is used. They are typically found in news [8] and advertisement [12] recommender systems. In the last category of sequential recommenders – session-aware recommendations, we have information about both the last interaction of the user with the system, as well as the history of the user. These recommenders are often implemented in e-commerce or for app suggestions [10, 14, 27]. In our work, we use the last method to approach sequential group recommendations. The above classification has been done for single user recommenders, and to our knowledge, our work is the first one in sequential group recommendations, that handles the notion of fairness.

3 BASIC CONCEPTS

In this section, we introduce a simple group recommendation process. This is a stand-alone process and does not take into account any past interactions that the group may have had with the system. The group recommendation approach we use, is the aggregation of the individual group members' preference lists into one group preference list. We define a preference list to be the list we recommend either to each member individually or to the group.

Specifically, let U be a set of users and I be a set of items. Each user $u_i \in U$ has given a rating from 1 to 5 to an item $d_z \in I$ represented as $r(u_i, d_z)$ The subset of users that rated an item $d_z \in I$ is denoted by $U(d_z)$, while the subset of items rated by a user $u_i \in U$ is denoted by $I(u_i)$. Let's assume also a fixed group of users G of size n, $G \subseteq U$. Having applied a single user recommendation algorithm to all group members in G and produced their preference lists $A_{u_1}, \ldots, A_{u_n}$, the next step is to aggregate these lists into one. After the aggregation phase, each item will have just one group preference score and the top k will be reported back to the group as the final recommendations.

Typically, two well established aggregation methods are used [2]. The first aggregation approach we examine is the *average* method. The main idea behind this approach is that all members are considered equals. So, the group preference of an item will be given be averaging its scores across all group members:

$$avgG(d_z, G) = \frac{sum_{u_i \in G} p(u_i, d_z)}{|G|},$$

where $p(u_i, d_z)$ gives us the preference score of d_z for user u_i (computed by a standard single user recommendation algorithm). The second aggregation method is *least misery*, where one member can act as a veto for the rest of the group. In this case, the group preference score of an item d_z is the minimum score assigned to that item in all group members preference lists:

$$minG(d_z, G) = min_{u_i \in G} p(u_i, d_z).$$

4 SEQUENTIAL GROUP RECOMMENDATIONS

In this section, we introduce the notion of multiple rounds to the previous group recommendation process. Consequently, we do not consider each group query to the system as a stand alone process, but as a sequence of queries submitted to the system by the same group. We call each group query to the system an *iteration*, and each iteration has the main components of a standard group recommendation method: single user recommendations followed by the aggregation phase.

Formally, let $\mathcal{GR}$ be a sequence of μ group recommendations $(Gr_1, \ldots, Gr_\mu)$. We use $p_j(u_i, d_z)$ for the preference score of user u_i for item d_z at iteration j, $1 \leq j \leq \mu$. These scores are estimated by a single user recommendation algorithm. After the aggregation phase of the group recommender system, we use $gp_j(G, d_z)$ to denote the preference score of item d_z for the group G as a whole, as estimated by the group recommender at iteration j.

This new model, in contrast to the previous one, introduces the notion of *multiple iterations* or *sequence* of recommendations. We use these interactions, to alter the output of the recommender system, in such a way that if a user was not satisfied in a previous iteration, potentially, he/she will be satisfied during the current one.

4.1 Satisfaction Measure

To examine the effectiveness of our group recommender algorithm through a series of iterations, we need to define a measure that will help us elevate the problem from examining each iteration independently, to examining a series of iterations. We introduce the notion of *satisfaction* that represents the gratification of each group member for the recommended items after each iteration of the system. We define two variations of satisfaction: *single user satisfaction* and *group satisfaction.*

4.1.1 *Single User Satisfaction.*

First, we want to provide a formal measure of the degree of the satisfaction of each user u_i in G to the group recommendation Gr_j received at step j. We do so, by comparing the quality of recommendations that the user receives as a member of the group with the quality of the recommendations that the user would have received as an individual.

Assume that Gr_j involves items $i_{j,1}, i_{j,2}, \ldots i_{j,k}$. Furthermore, let $A_{u_i,j}$ be the list with the top-k items $a_{j,1}, a_{j,2}, \ldots a_{j,k}$ for user u_i, that is, the k items with the highest prediction scores for user u_i. Our goal is to directly compare the user's satisfaction from the group recommendation list with the ideal case for that user. This gives us a more clear view of the satisfaction of the user than the average method since we take into account the top items for each user, and not only the top items for the group. Formally:

$$sat(u_i, Gr_j) = \frac{GroupListSat(u_i, Gr_j)}{UserListSat(u_i, A_{u_i,j})} \tag{1}$$

$$GroupListSat(u_i, Gr_j) = \sum_{d_z \in Gr_j} p_j(u_i, d_z) \tag{2}$$

$$UserListSat(u_i, A_{u_i,j}) = \sum_{d_z \in A_{u_i,j}} p_j(u_i, d_z) \tag{3}$$

With the function *GroupListSat* (Equation 2), we calculate the user's satisfaction based on the group recommendation list. For every item in Gr_j, we sum the score as they appear in each user's $A_{u_i,j}$. The function *UserListSat* (Equation 3), calculates the ideal case for the user, by simply sum the scores of the k top items in the user's $A_{u_i,j}$. This way, we are able to normalize the user's satisfaction score. For example, if a user gives mainly low scores to items, then Equation 1 is able to compensate for it.

Note that in both Equations 2 and 3, we do not use the scores as they appear in the group list, but as they appear in the individual preference list of the user. Since the aggregation phase of the group recommendation process, rather distorts the individual opinions of the group members, we opt to take into consideration only the personal preference scores of each group member.

Still, Equation 1 remains *static*, in the sense that we calculate the satisfaction of a user for one iteration only. As stated before, what we want is to define a measure that takes into consideration the satisfaction scores from previous iterations of the system. Such a score will represent the overall satisfaction of each group member with the entirety of the μ group recommendations.

Definition 4.1 (Overall Satisfaction). The overall satisfaction of user u_i with respect to a sequence $\mathcal{GR}$ of μ iterations is the average of the satisfaction scores after each iteration:

$$satO(u_i, \mathcal{GR}) = \frac{\sum_{j=1}^{\mu} sat(u_i, Gr_j)}{\mu} \tag{4}$$

4.1.2 Group Satisfaction. Having defined the satisfaction score of each group member we can now define the satisfaction score of the entire group. Specifically, the satisfaction of the group G with respect to a group recommendation list Gr_j is defined as the average of the satisfaction of the users in the group:

$$groupSat(G, Gr_j) = \frac{\sum_{u_i \in G} sat(u_i, Gr_j)}{|G|} \tag{5}$$

Subsequently, we define the overall group satisfaction of a group G for a recommendation sequence $\mathcal{GR}$ of μ group recommendations $(Gr_1, \ldots, Gr_\mu)$, as:

$$groupSatO(G, \mathcal{GR}) = \frac{\sum_{u_i \in G} satO(u_i, \mathcal{GR})}{|G|} \tag{6}$$

This measure indicates if the items we report to the group, are acceptable to its members. Higher group satisfaction means that the group members are satisfied with the recommendations. However, since we average the members satisfaction scores a dissatisfaction of a user can probably be lost in the computations. To counter this problem, we focus as well on the potential disagreements between the users in the group. For representing such *disagreement*, we define the *groupDis* measure:

$$\begin{aligned} &groupDis(G, \mathcal{GR}) = \\ &max_{u_i \in G} satO(u_i, \mathcal{GR}) - min_{u_i \in G} satO(u_i, \mathcal{GR}) \end{aligned} \tag{7}$$

Intuitively, we define the disagreement of the group, to be the difference in the overall satisfaction scores between the most satisfied and the least satisfied member in the group. Ideally, we want this measure to take low values, as that will indicate that the group members are all satisfied to the same degree. Higher *groupDis* values will demonstrate that at least one member of the group is biased against.

4.2 Problem Definition

Our sequential group recommender system needs to achieve two independent objectives that are nonetheless critical to the success of our model. The first objective considers the group as an entity: we want to offer to the group the best possible results. The second objective considers the group members independently: we want to behave as fairly as possible towards all members.

Definition 4.2 (Fair Sequential Group Recommendation). Given a group G, the sequential group recommender produces at the μ iteration a list of k items Gr_μ, $Gr_\mu \in \mathcal{GR}$ that:

(1) Maximizes the overall group satisfaction, $groupSatO(G, \mathcal{GR})$, and
(2) Minimizes the variance between users satisfaction scores: $groupDis(G, \mathcal{GR})$.

To achieve the first objective, we target at maximizing *groupSatO*. This means that we require the items with the highest preference scores for the group as a whole. Since *groupSatO* expresses the average satisfaction of the group members, and the dissatisfaction of just one member is easily lost, we also need to achieve the second objective, to minimize *groupDis*, which is the representation of the degree of dissatisfaction between the members of the group.

Depending on the similarity between the group members, these two objectives may be conflicting with each other. A simple example, is a group of three members, two that are highly similar to each other, and one that is very dissimilar to them. To achieve high *groupSatO* values, we need to recommend items that are relevant to the two similar users, so as to increase the average satisfaction of the group. On the other hand, by doing so, *groupDis* will take high values as well since we do not address the needs of the third member. Overall, we need to recommend items that are not just good enough for all members – since that still returns low *groupSatO* values, but items that have high relevance to the group, without sacrificing the opinions of the minority.

4.3 Sequential Hybrid Aggregation Method

As explained above, both the *average* and the *least misery* aggregation methods have drawbacks, when we consider them for sequential recommendations. At the same time, both have advantages (namely, equality for average, and inclusion of all opinions for least misery) that are an asset for a recommender system. Furthermore, it has been proven that to compute the best suggestions for a group that minimizes the gap between the least and highest satisfied group members is an NP-Hard problem [35]. That is something we want to achieve throughout the multiple iterations of the system, to be equally fair to all members of the group. To circumvent the problem and to capitalize on the advantages of the average and the least misery aggregation methods, we propose a novel aggregation method that is a weighted combination of them. We call the method *sequential hybrid aggregation method.*

$$\begin{aligned} &score(G, d_z, j) = \\ &(1 - \alpha_j) * avgScore(G, d_z, j) + \alpha_j * leastScore(G, d_z, j) \end{aligned} \tag{8}$$

The function $avgScore(G, d_z, j)$ returns the score of the item d_z as it is computed by the average aggregation method during iteration j, and function $leastScore(G, d_z, j)$ returns the least satisfied user's score of d_z at iteration j. The variable α takes values from 0 to 1. If $\alpha = 0$, then the sequential hybrid aggregation method becomes average aggregation, while when $\alpha = 1$, it transforms to a modified least misery aggregation, where we only take into account the preferences of the least satisfied member.

Intuitively, when $\alpha = 0$, our method satisfies the first part of the fair sequential group recommendation problem (Definition 4.2) since the average aggregation considers the best options for the group as a whole. The benefits of $\alpha = 1$ can be seen for higher values of μ, since at the first iterations the group disagreement may remain high since it considers the overall satisfaction of the users (Equation 4).

Given that our goal is to fulfill both objectives of the problem definition, we need to set the value of α between 0 and 1. Furthermore, we want this value to self-regulate, to more effectively describe the consensus of the group. Thus, we set the value of α dynamically in each iteration by subtracting the minimum satisfaction score of the group members in the previous iteration, from the maximum score.

$$\alpha_j = max_{u \in G} sat(u, Gr_{j-1}) - min_{u \in G} sat(u, Gr_{j-1}) \quad (9)$$

When $j = 1$ (the first iteration of the system), then $\alpha = 0$, and the aggregation method reverts to that of a classic average aggregation.

By having this dynamically calculated α, we counteract the individual drawbacks of average and least misery. Intuitively, if the group members are equally satisfied at the last round, then α takes low values, and the aggregation will closely follow that of an average, where everyone is treated as an equal. On the other hand, if one group member is extremely unsatisfied in a specific iteration, then α takes a high value and promotes that member's preferences on the next iteration. Formally:

PROPOSITION 1. *Let u be a user in a group G, such that, $sat(u, Gr_{j-1}) < sat(u_i, Gr_{j-1})$, $\forall u_i \in G \backslash \{u\}$. Then, $\exists u_y \in G \backslash \{u\}$, such that, $sat(u_y, Gr_j) < sat(u, Gr_j)$.*

PROOF. For the purpose of contradiction, assume that $sat(u, Gr_j) < sat(u_y, Gr_j)$, $\forall u_y \in G \backslash \{u\}$. Then, this means that u is the least satisfied user at iteration j, which in turn means that α takes values not leading towards a least misery approach at this iteration, meaning that u was not the least satisfied user at iteration $j - 1$, which is a contradiction. □

To exemplify this, we run the same experiment as in Figure 1, for the same group, but now we utilize the sequential hybrid aggregation method. The results appear in Figure 2. Here, we can observe that a group member that was not satisfied in the previous iteration of the system, is satisfied in the next. A good example of this is User 4 where in the first iteration has a very low satisfaction score, and in the second has a higher one. This is a clear improvement over the results shown in Figure 1, where User 4 was always the least satisfied member of the group.

4.4 Algorithm

Our sequential hybrid group aggregation method is described in Algorithm 1. The input to the algorithm is the group G, the iteration j, and the size k of the group recommendation list Gr_j, which we report to the group after each iteration is finished. In Line 1, we apply a single user recommendation algorithm to each group member and save their preference lists into the variable A. In Line 2, we define a set that contains all the items that appear in the members' preference lists Gl and, in Lines 3–5, we populate the set. In Lines 6–10, we calculate the α; if it is the first iteration meaning $j = 1$ (Line 7), then $\alpha = 0$, otherwise, we use Equation 9 (Line 9) to calculate it. In Lines 11–24, we perform the sequential hybrid aggregation method. For all the items in Gl, we calculate the item score using Equation 8 (Line 12) and insert them in the group list Gr_j (Line 13). After we have examined all the items in Gl, we sort the items in the group list (Line 15), and finally, we report the top-k of them to the group (Line 16). The complexity of our algorithm is $O(n \log n)$ due to the time complexity of the sorting function, where $n = |Gl|$.

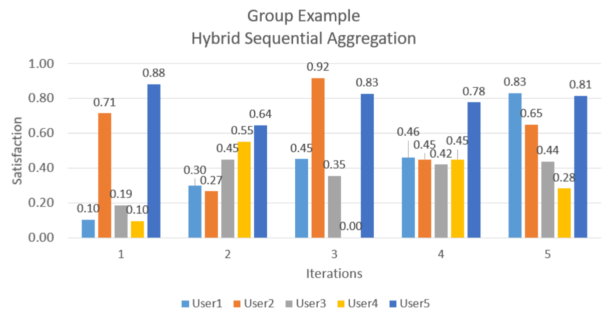


Figure 2: Example for the same group as in Figure 1. We consider 5 rounds. We utilize the Sequential Hybrid Aggregation method.

```
   Data: Group G, $j$ top $k$
   Result: Group Recommendation List: $Gr_j$
1  $A \leftarrow RS(G)$;
2  $Gl \leftarrow \emptyset$;
3  for $u_i \in G$ do
4      $Gl \leftarrow Gl \cup A_{u_i}$;
5  end
6  if $j$=1 then
7      $\alpha_j = 0$;
8  else
9      $\alpha_j = maxSat(G, j-1) - minSat(G, j-1)$;
10 end
11 for item $d_z$ in $Gl$ do
12     $score(G, d_z, j) =$
         $(1 - \alpha_j) * avgScore(G, d_z, j) + \alpha_j * leastScore(G, d_z, j)$;
13     $Gr_j \leftarrow Gr_j \cup score(G, d_z, j)$;
14 end
15 $sort(Gr_j)$;
16 $report(top(Gr_j, k))$;
```
Algorithm 1: Sequential Group Recommendation Algorithm

5 EXPERIMENTAL SETUP

In this section, we describe the two pre-processing tasks needed for the experimental evaluation of our sequential group recommender system: the dataset splitting and the group formation.

5.1 Dataset

For the experimental evaluation of our sequential group recommender system, we use the 20M MovieLens Dataset [11]. It contains 20.000.263 ratings across 27.278 movies. These data were created by 138.493 users between January 09, 1995, and March 31, 2015. The dataset was generated on March 31, 2015. To simulate multiple iterations of suggestions, we split the dataset into chunks. Each chunk is added to the system, representing new information, and along with the already existing data in the system, is used for locating the suggestions for the next iteration.

Initially, we divide the dataset into two parts of roughly the same size. The first part that contains the ratings given between January 1994 and December 2003, is the starting dataset of our recommender. This gives us an initial dataset that consists of 8.381.255 ratings, 73.519 users and 6.382 movies. We initiate the system with this starting dataset to avoid any issues emanating from the cold start problem[2]. The second part of the dataset is further split into chunks based on timestamps. We create 22 chunks, where each one includes information for a period of six months. During the first iteration, the system will have access only to the initial dataset. When that iteration ends, the system will be enhanced with one additional chunk (first semester of 2004), and after each subsequent iteration, the system will be given access to the next chunk (second semester of 2004, fist semester of 2005, etc.).

Figure 3 shows the detailed information on the contents of each chunk. With the last two columns, we report the number of users and movies respectively that appear for the first time during the specific semester. As we can see, the ratings are not evenly split between the semesters. There is a higher number of ratings in the years 2004 to 2010, in some cases even triple the amount. As it is expected, the number of movies that appear in each chunk increases as the time passes, since the users can rate old movies as well. At the same time, the number of users is relatively the same across all chunks. The disinterested of the users to give ratings as time passes is not a complication in our evaluations since the total number of available ratings is enough for our needs.

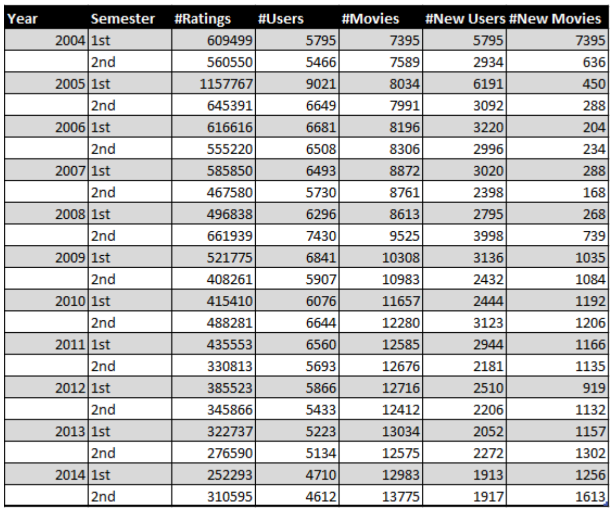
5.2 Group Formation

We will evaluate our sequential group recommender on stable groups. This means that the members of the group do not change between iterations, and all the members are present for each recommendation. We will examine our recommender on four types of groups, based on the similarity shared between the members. We target at covering groups with different semantics, starting from groups with similar users to groups with dissimilar ones. We calculate the similarity between the members of the group using only the starting dataset and not the entirety of it since we want the relationship between the users to be present from the first iteration.

The similarity between the group members is calculated using the Pearson Correlation similarity measure [28], which takes values from −1 to 1. Higher values imply a higher similarity between the users, while negative values indicate dissimilarity.

$$s(u_i, u_l) = \frac{\sum\limits_{d_z \in X} (r(u_i, d_z) - \bar{r}_{u_i})(r(u_l, d_z) - \bar{r}_{u_l})}{\sqrt{\sum\limits_{d_z \in X} (r(u_i, d_z) - \bar{r}_{u_i})^2} \sqrt{\sum\limits_{d_z \in X} (r(u_l, d_z) - \bar{r}_{u_l})^2}} \quad (10)$$

[2]The cold start problem in collaborative filtering appears when there is not enough information about a user and we are unable to find similar other users to him/her. This problem is beyond the scope of this research, and to overcome it, we initialize our system with a large enough dataset.



Year	Semester	#Ratings	#Users	#Movies	#New Users	#New Movies
2004	1st	609499	5795	7395	5795	7395
	2nd	560550	5466	7589	2934	636
2005	1st	1157767	9021	8034	6191	450
	2nd	645391	6649	7991	3092	288
2006	1st	616616	6681	8196	3220	204
	2nd	555220	6508	8306	2996	234
2007	1st	585850	6493	8872	3020	288
	2nd	467580	5730	8761	2398	168
2008	1st	496838	6296	8613	2795	268
	2nd	661939	7430	9525	3998	739
2009	1st	521775	6841	10308	3136	1035
	2nd	408261	5907	10983	2432	1084
2010	1st	415410	6076	11657	2444	1192
	2nd	488281	6644	12280	3123	1206
2011	1st	435553	6560	12585	2944	1166
	2nd	330813	5693	12676	2181	1135
2012	1st	385523	5866	12716	2510	919
	2nd	345866	5433	12412	2206	1132
2013	1st	322737	5223	13034	2052	1157
	2nd	276590	5134	12575	2272	1302
2014	1st	252293	4710	12983	1913	1256
	2nd	310595	4612	13775	1917	1613

Figure 3: Divided by semester: Ratings, unique users and movies, as well as the number of new users and movies that appear in each chunk

where $X = I(u_i) \cap I(u_l)$ and $\bar{r}_{u_i}$ is the mean of the ratings in $I(u_i)$, i.e., the mean of the ratings of user u_i.

We consider two users to be highly similar to each other, if they share above 0.5 similarity score, while dissimilar when they have -0.5 or lower similarity score. The types of groups, we are considering are the following:

- **4 similar – 1 dissimilar:** The four members of the group share a high similarity score, while the dissimilar one shares a low similarity score with the rest of the group members.
- **3 similar – 2 similar:** We divide the group into two subgroups. The members of each subgroup are similar to each other, while at the same time, the subgroups are dissimilar to one another, i.e., all members of one subgroup are dissimilar to all members of the other subgroup.
- **3 similar – 1 dissimilar – 1 dissimilar:** We divide the group into three separate subgroups: one that contains three members and two subgroups that each contain just one user. All subgroups are dissimilar to each other, while the three members belonging in the same subgroup are similar to each other.
- **5 dissimilar:** All members of the group are dissimilar with each other.

For each group type, we generate 100 groups, and each user can only participate in one group per category. We consider only groups of five members since that is a realistic scenario.

6 EVALUATION

In this section, we go over our experimental procedure and the variables we took into consideration during the evaluation of the sequential group recommender. We present the results of this evaluation that showcase the advantages of our sequential hybrid aggregation method.

6.1 Experimental Procedure

In our experiments, we will examine the behavior of four types of groups, considering 100 different groups per type. For each group in a set, we perform sequential group recommendations for various values of μ, as described in Algorithm 1. For all the groups in the set, we calculate the average of $groupSatO(G, \mathcal{GR})$ and $groupDis(G, \mathcal{GR})$. Additionally, we utilize an F-score measure (Equation 11), namely the harmonic mean of the *groupSatO* and *groupDis* measures, that provides a good indication of the users' satisfaction and the agreements between the users in the group. Taking into account the input functions that F-score needs, we use $1 - groupDis$ to simulate the group agreement.

$$F\text{-}score = 2\frac{groupSatO * (1 - groupDis)}{groupSatO + (1 - groupDis)} \tag{11}$$

We find similarities between users, by utilizing the Pearson Correlation. We consider two users as similar if the similarity is greater than 0.7, and if they have rated more than five identical items.

To predict preference scores for a user, we use the Weighted Sum of Others Ratings [33], and we take into account only the top 100 most similar users to him/her. We recommend to the group the 10 items with the highest group preference score. In our case, we assume that the system does not recommend items to the group that it has previously recommended.

$$p(u_i, d_z) = \bar{r}_{u_i} + \frac{\sum_{u_l \in (P_{u_i} \cap U(d_z))} s(u_i, u_l)(r(u_l, d_z) - \bar{r}_{u_l})}{\sum_{u_l \in (P_{u_i} \cap U(d_z))} |s(u_i, u_l)|} \tag{12}$$

6.2 α Experiments

To examine the behavior of the sequential hybrid aggregation method, we first examine the values that the variable α (Equation 9) takes during the iterations of the system. We examine the performance of α through 15 iterations. In Figure 4, we report the average α values of the 100 groups per type, per iteration. At the first iteration, the α takes the default value 0. During the next iterations, the α values increase in a close to linear form. This is expected since by design the groups have at least one outlier member that is dissimilar to the rest of the group. This guarantees that at least one member is not satisfied during an iteration of the system. The high values of α in the later iterations are also the result of the dataset splitting. At the later iterations, the majority of the dataset has already been given as input to the system, and statistically, the best items for the group as well as for the individual members have already been recommended. Since per our scenario our system cannot suggest items that have already been recommended, the remaining items are less preferable to the group as a whole and each member individually.

An additional observation is the different behaviors of the four group types. As already stated, the groups are formed in such a way, as to always have at least one member being a minority. The α values offer a first impression of the implications these types of groups have on sequential recommendations. Remember that α is different than the group disagreement (Equation 7) since it only takes into account the members' satisfaction for the previous iteration of the system. During the first iterations of the system, there is a little variation in the values of α. The clear distinction becomes obvious in the later iterations, where we can observe that

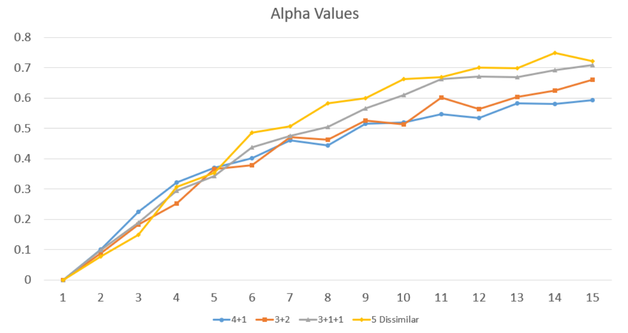


Figure 4: Average α values over 100 groups per group type, at each iteration.

the more diverse the group becomes, the more higher the values of α get. This is expected since the more distinct opinions in the group are, the higher the disagreement between the user becomes, and thus the values of α become higher.

6.3 Group Type Experiments

To examine the effectiveness of the sequential group recommender, we consider the group satisfaction, *groupSatO* (Equation 6) and the disagreement between the users, *groupDis* (Equation 7), after a number of iterations μ. For each type of group, we perform the experiments for different number of iterations, $\mu \in [5, 10, 15]$. Figures 5, 6, 7 and 8, show the average of the group satisfaction and group disagreement scores for the 100 groups per group type, respectively.

We perform the tests for different values of α, $\alpha \in [0, 0.4, 1, dynamic\ \alpha]$, as well as the pure Least Misery approach (LM). For $\alpha = 0$, we have a standard average aggregation method. For $\alpha = 1$, we have a slightly altered least misery aggregation method, where instead of assigning the lowest score among the individual group members' preference scores to the item as a group preference score, we assign to the item the score as it appears on the preference list of the least satisfied user. We also consider a static value for α, which is 0.4, selected as the one with the best results, after extensive experimentation on different α values.

When considering the overall group satisfaction, we observe that for all group types it decreases by the same degree. This is because in the later iterations of the system the best items for the group have already been reported, and as stated we do not recommend items that have already been recommended in previous iterations. When comparing the group satisfaction for the different values of α, we observe that the average method slightly outperforms the dynamic α. For 15 iterations, the average method offers a better overall group satisfaction by a slight factor, since it tries to offer the best results to the group as a whole. The dynamic α, also clearly outperform the LM method, especially the more diverse the groups become. By far the worst performance is for $\alpha = 1$, since in each iteration of the system, we consider only one user, without taking into account the opinions of the rest of the group.

Regarding the group disagreement, LM has the worst results followed by the average method ($\alpha = 0$), while the dynamic α has the best. This is because, the average method ignores the minority

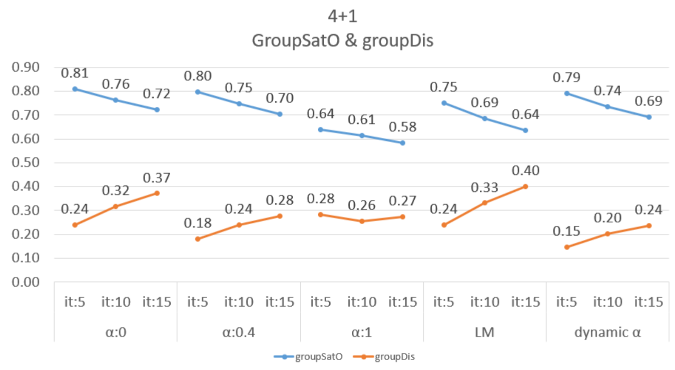


Figure 5: 4 similar – 1 dissimilar: Overall group satisfaction and group disagreement for 100 groups for different μ.

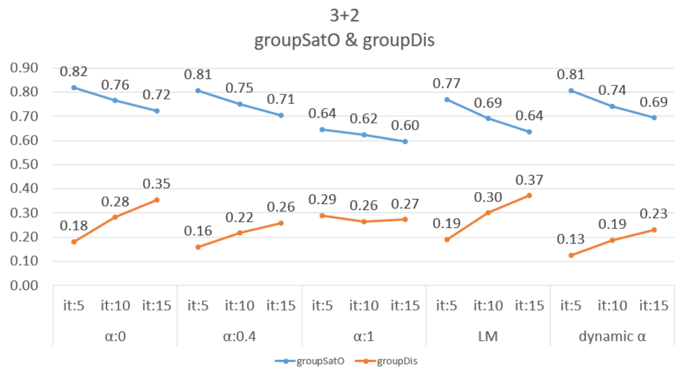


Figure 6: 3 similar – 2 similar: Overall group satisfaction and group disagreement for 100 groups for different μ.

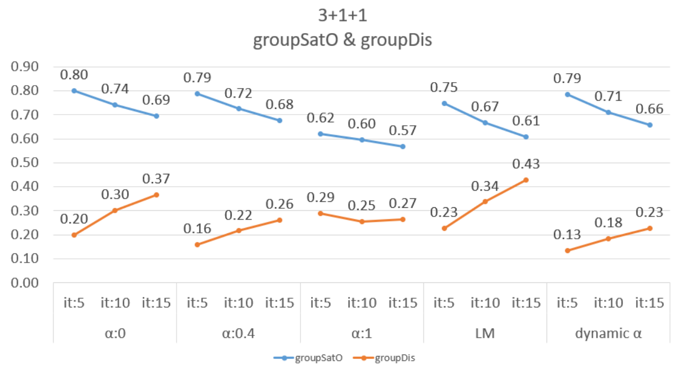


Figure 7: 3 similar – 1 dissimilar – 1 dissimilar: Overall group satisfaction and group disagreement for 100 groups for different μ.

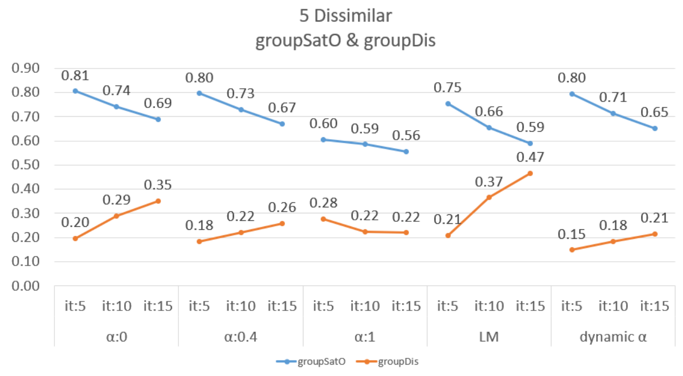


Figure 8: 5 dissimilar: Overall group satisfaction and group disagreement for 100 groups for different μ.

opinion in the group, which is reflected in the high group disagreement values. LM does not work well for diverse groups since the preferences of the members greatly differ from each other. This becomes more apparent the more dissimilar the group members are, as shown in Figure 8. The benefits and drawbacks of $\alpha = 1$ are apparent in our evaluations. We can observe high disagreement in the first iterations of the system, but later, we have better results. This is more clear in more diverse groups, such as the ones in Figure 8, where during the later iterations it achieves the second best disagreement.

Overall, the sequential group recommendations problem is not one dimensional. We need to examine the overall group satisfaction and the group disagreement in conjuncture with each other. Even though the average method ($\alpha = 0$) and the static value of $\alpha = 0.4$, offers slightly better group satisfaction than the dynamic α, they both have far higher group disagreement values. We argue that this is an acceptable loss of group satisfaction, when we consider the advantage that the dynamic α has over the average and static α aggregation methods on the disagreement between the group members.

To better demonstrate this, we also calculate the F-score of the *groupSatO* and *groupDis* values. We present these results in Table 1. For all group types and all number of iterations, the method using dynamic α offers better results, than the rest of the methods. This better demonstrates the point we made previously. Even if the average method offers better group satisfaction, our proposed dynamic hybrid sequential method, more than makes up for it, with the far lower group disagreement.

7 CONCLUSION

In this work, we introduce the notion of sequential group recommendation. We show that the standard group recommendation approaches are not favorable for sequential recommendations. Such methods suffer from particular drawbacks – a minority opinion may be lost by the average method, and the preference of the group is determined by just one voice in the least misery – that are only exacerbated when they are applied to sequential recommendations. We propose a sequential group recommendation model that takes into account the satisfaction of the members during the previous interactions of the group with the system. The influence that each member has on determining the group score for an item, is defined by the degree of the satisfaction that member has, for the recommended items in the previous iteration of the system. In the future,

Group	4+1			3+2			3+1+1			5Diss		
Iterations	5	10	15	5	10	15	5	10	15	5	10	15
$\alpha = 0$	0.784	0.720	0.672	0.819	0.741	0.682	0.801	0.719	0.662	0.805	0.726	0.668
$\alpha = 0.4$	0.808	0.754	0.713	0.823	0.767	0.724	0.814	0.753	0.706	0.806	0.753	0.705
$\alpha = 1$	0.675	0.673	0.648	0.676	0.676	0.655	0.662	0.663	0.640	0.658	0.667	0.649
LM	0.756	0.677	0.617	0.789	0.695	0.633	0.760	0.664	0.590	0.771	0.645	0.560
Dynamic α	**0.821**	**0.765**	**0.725**	**0.839**	**0.775**	**0.730**	**0.824**	**0.760**	**0.712**	**0.823**	**0.762**	**0.712**

Table 1: F-score values for all group categories per α value, per iteration.

we want to further evaluate our work with a user study, to better examine the effectiveness of our method and how users respond to it. Additionally, we want to study the effect of sequential group recommendations on ephemeral groups and the subsequent consequence of them on the individual members. That is, how we can satisfy a user if at each iteration is a member of a different group, without sacrificing the overall satisfaction of the temporally formed group.

ACKNOWLEDGEMENTS

This research was partially supported by the project *Ethical AI for the Governance of Society* (ETAIROS, grant #327352) funded by the Academy of Finland.

REFERENCES

[1] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawlat, Gautam Das, and Cong Yu. 2009. Group Recommendation: Semantics and Efficiency. *PVLDB* 2, 1 (2009), 754–765.

[2] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. 2010. Group Recommendations with Rank Aggregation and Collaborative Filtering. In *RecSys*.

[3] Fidel Cacheda, Víctor Carneiro, Diego Fernández, and Vreixo Formoso. 2011. Comparison of Collaborative Filtering Algorithms: Limitations of Current Techniques and Proposals for Scalable, High-performance Recommender Systems. *ACM Trans. Web* 5, 1 (2011), 2:1–2:33.

[4] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang, and Richang Hong. 2018. Attentive Group Recommendation. In *The 41st International ACM SIGIR Conference on Research &#38; Development in Information Retrieval (SIGIR '18)*. https://doi.org/10.1145/3209978.3209998

[5] Lucas Augusto Montalvão Costa Carvalho and Hendrik Teixeira Macedo. 2013. Users' Satisfaction in Recommendation Systems for Groups: An Approach Based on Noncooperative Games. In *Proceedings of the 22Nd International Conference on World Wide Web (WWW '13 Companion)*. https://doi.org/10.1145/2487788.2488090

[6] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where You Like to Go Next: Successive Point-of-Interest Recommendation. In *International Joint Conference on Artificial Intelligence*.

[7] Zunping Cheng and Neil Hurley. 2009. Effective diverse and obfuscated attacks on model-based recommender systems. In *RecSys*.

[8] Florent Garcin, Christos Dimitrakakis, and Boi Faltings. 2013. Personalized News Recommendation with Context Trees. *CoRR* abs/1303.0665 (2013).

[9] Francesca Guzzi, Francesco Ricci, and Robin Burke. 2011. Interactive Multi-party Critiquing for Group Recommendation. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*. https://doi.org/10.1145/2043932.2043980

[10] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-aware Music Recommendation Based on Latenttopic Sequential Patterns. In *RecSys*.

[11] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4 (2015), 19:1–19:19.

[12] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *RecSys*.

[13] Anthony Jameson and Barry Smyth. 2007. The Adaptive Web. Springer-Verlag, Berlin, Heidelberg, Chapter Recommendation to Groups, 596–627.

[14] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. 2015. Adaptation and Evaluation of Recommendations for Short-term Shopping Goals. In *RecSys*.

[15] Kai Yu, A. Schwaighofer, V. Tresp, Xiaowei Xu, and H. . Kriegel. 2004. Probabilistic memory-based collaborative filtering. *IEEE TKDE* 16, 1 (2004), 56–69.

[16] Jae Kyeong Kim, Hyea Kyeong Kim, Hee Young Oh, and Young U. Ryu. 2010. A group recommendation system for online communities. *International Journal of Information Management* (2010). https://doi.org/10.1016/j.ijinfomgt.2009.09.006

[17] Defu Lian, Vincent W. Zheng, and Xing Xie. 2013. Collaborative Filtering Meets Next Check-in Location Prediction. In *WWW*.

[18] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the Next Location: A Recurrent Model with Spatial and Temporal Contexts. In *AAAI Conference on Artificial Intelligence*.

[19] Lucas Machado and Kostas Stefanidis. 2019. Fair Team Recommendations for Multidisciplinary Projects. In *WI*.

[20] Lihi Naamani Dery, Meir Kalech, Lior Rokach, and Bracha Shapira. 2010. Iterative Voting Under Uncertainty for Group Recommender Systems. In *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10)*. https://doi.org/10.1145/1864708.1864763

[21] Eirini Ntoutsi, Kostas Stefanidis, Kjetil Nørvåg, and Hans-Peter Kriegel. 2012. Fast Group Recommendations by Applying User Clustering. In *ER*.

[22] Eirini Ntoutsi, Kostas Stefanidis, Katharina Rausch, and Hans-Peter Kriegel. 2014. "Strength Lies in Differences": Diversifying Friends for Recommendations through Subspace Clustering. In *CIKM*.

[23] Mark O'Connor, Dan Cosley, Joseph A. Konstan, and John Riedl. 2001. PolyLens: A recommender system for groups of user. In *Proceedings of the Seventh European Conference on Computer Supported Cooperative Work*.

[24] Michael J. Pazzani and Daniel Billsus. 2007. *Content-Based Recommendation Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 325–341.

[25] D. Qin, X. Zhou, L. Chen, G. Huang, and Y. Zhang. 2018. Dynamic Connection-based Social Group Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2018). https://doi.org/10.1109/TKDE.2018.2879658

[26] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *ACM Comput. Surv.* 51, 4 (2018), 66:1–66:36.

[27] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *RecSys*.

[28] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *CSCW*.

[29] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. *Collaborative Filtering Recommender Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 291–324.

[30] Dimitris Serbos, Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2017. Fairness in Package-to-Group Recommendations. In *WWW*.

[31] Maria Stratigi, Haridimos Kondylakis, and Kostas Stefanidis. 2017. Fairness in Group Recommendations in the Health Domain. In *ICDE*.

[32] Maria Stratigi, Haridimos Kondylakis, and Kostas Stefanidis. 2018. FairGRecs: Fair Group Recommendations by Exploiting Personal Health Information. In *DEXA*.

[33] Taghi M.; Su, Xiaoyuan;Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence* 2009 (2009). https://doi.org/10.1155/2009/421425

[34] John Von Neumann and Oskar Morgenstern. 1947. *Theory of games and economic behavior, 2nd rev. ed.* Princeton University Press.

[35] Lin Xiao, Zhang Min, Zhang Yongfeng, Gu Zhaoquan, Liu Yiqun, and Ma Shaoping. 2017. Fairness-Aware Group Recommendation with Pareto-Efficiency. In *RecSys*.

[36] Quan Yuan, Gao Cong, and Chin-Yew Lin. 2014. COM: A Generative Model for Group Recommendation. In *KDD*.