Noname manuscript No.
(will be inserted by the editor)

ART: Group Recommendation Approaches for Automatically Detected Groups

Ludovico Boratto · Salvatore Carta

Received: date / Accepted: date

Abstract Group recommender systems provide suggestions when more than a person is involved in the recommendation process. A particular context in which group recommendation is useful is when the number of recommendation lists that can be generated is limited (i.e., it is not possible to suggest a list of items to each user). In such a case, grouping users and producing recommendations to groups becomes necessary. None of the approaches in the literature is able to automatically group the users in order to overcome the previously presented limitation. This paper presents a set of group recommender systems that automatically detect groups of users by clustering them, in order to respect a constraint on the maximum number of recommendation lists that can be produced. The proposed systems have been largely evaluated on two real-world datasets and compared with hundreds of experiments and statistical tests, in order to validate the results. Moreover, we introduce a set of best practices that help in the development of group recommender systems in this context.

Keywords Group Recommendation · Clustering · Group Modeling

1 Introduction

Recommender systems suggest items that might be interesting for a user [45]. In its general meaning, *group recommendation* is designed for contexts in which more than one person is involved in the recommendation process [6, 21]. An

This work is partially funded by Regione Sardegna under project SocialGlue, through PIA - Pacchetti Integrati di Agevolazione "Industria Artigianato e Servizi" (annualità 2010), and by MIUR PRIN 2010-11 under project "Security Horizons".

Dipartimento di Matematica e Informatica - Università di Cagliari
Via Ospedale 72 - 09124 Cagliari, Italy
E-mail: {ludovico.boratto,salvatore}@unica.it

application scenario in which group recommendations have to be produced is the following:

> *A group of people decides to dine together. In order to choose the restaurant in which they should go, each participant expresses her/his preferences about different types of food. These preferences have to be combined, in order to choose a restaurant that maximizes the group satisfaction (i.e., that has a type of food that satisfies most of the group).*

A particular application scenario in which group recommendation can be used is when the number of recommendation lists that can be produced is limited (i.e., it is not possible to recommend a list of items to each user). The example below presents a case in which there is such a constraint.

> *A company decides to print recommendation flyers that present suggested products. Even if the data to produce a flyer that contains individual recommendations for each customer is available, the process of printing a different flyer for everyone would be technically too hard to accomplish and costs would be too high. A possible solution would be to set a number of different flyers to print, such that the printing process could be affordable in terms of costs and the recipients of the same flyer would be interested by its content.*

In the last few years, group recommendation has been highlighted as a challenge in the recommendation research [21,44]. With respect to classic group recommendation, the development of systems that deal with a limited number of recommendation lists is even more challenging, because of the additional problem of optimally defining groups, in order to respect the constraint and maximize user satisfaction. None of the approaches in the literature is able to automatically detect groups and deal with a scenario in which the number of recommendation lists that can be produced is limited, so *the definition of group recommender systems able to detect groups of users represents an open research problem.*

This paper presents *ART (Automatic Recommendation Technologies)*, a set of group recommender systems that also detect groups of users. In order to produce recommendations for a group that has been automatically detected, the proposed systems do not only cluster users and produce recommendations. In fact, this paper studies an approach to integrate individual predictions in the group detection task, in order to improve the quality of the groups (i.e., find users with more similar preferences). As this study will show, in order for a system to perform well, recommendation and classification need to be combined. Moreover, the different strategies to model the preferences of the detected groups and the predictions algorithms are tailored to this context.

Open problems and scientific contributions. The open problems in the development of a system that also detects groups are the following:

(a) Users have to be properly grouped, in order to respect the constraint on the number of recommendation lists that can be generated and to produce recommendations that are effective for the users in it.

(b) Once groups are formed, individual preferences have to be combined, in order to build a model that contains the preferences of a group.
(c) Group recommendations have to be generated, by either aggregating predictions built for each user, or by building specific predictions for a group.

The scientific contributions introduced to solve them are the following:

– Regarding point (a), the proposed solution for the automatic group detection is based on user *clustering*. In fact, in the considered context a partitioning in groups does not exist and a form of unsupervised classification is necessary. Producing recommendations for a set of similar users allows the system to satisfy individual users in a group and respect the constraint. In this paper, we study two approaches to cluster the users.
– Regarding point (b), J. Masthoff presented several studies related to *group modeling*, i.e., the process of combining multiple user models into a single model [29–34]. Different group modeling strategies exist, but none of them is useful in every context, so the choice of the strategy that best models a group should be made after analyzing the context in which a group is modeled [10, 43]. In this paper, several algorithms that implement the strategies to model automatically detected groups are proposed and compared.
– Regarding point (c), according to [21], a system can adopt three different approaches to rating prediction: (i) construction of group preference models and prediction of the missing ratings for each group by using the model; (ii) prediction of the ratings for the items not rated by each user and merging of all the individual recommendations made for the members of a group; (iii) aggregation of the predictions built for every user into a group preference. In this paper, we propose algorithms for rating prediction that implement these three approaches, in a tailored version specifically designed for a context in which groups are automatically detected. These algorithms have been tested and compared, in order to find the most accurate one.

The rest of the paper is organized as follows: Section 2 presents a survey on group recommendation; Section 3 contains a formal definition of the problem; Section 4 presents the proposed classes of systems; Section 5 illustrates the performed experiments; Section 6 contains conclusions and future work.

2 Group Recommendation: Background and Related Work

This section presents a survey of group recommendation, based on the different aspects that characterize a system.

2.1 Type of group

A group recommender system can operate in different contexts, which require to handle different types of group, listed below:

- a system has to provide recommendations to an *established group* of users who share the same interests and do something together;
- recommendations are provided to an heterogeneous and *occasional group* who has a common aim and shares the system on a particular occasion;
- a system recommends items in an environment shared by a *random group* of people who do not have anything in common.

The previous classification was introduced by us [6], and adopted in [11]. In the literature, a system works with established groups when recommending movies to persistent groups going to the cinema together [41], while occasional groups are considered when suggesting music in a gym [36], or to suggest tourist attractions to groups traveling together [2]; instead, random groups are employed when assisting groups of users in collaborative Web browsing [24], or when recommending news to a group of users that are in a public space [10].

2.2 Preference acquisition

A system can collect preferences considering the fact that a user is a part of a group, or not. Preference acquisition is tailored on the group in scenarios where the users can interact [37,38] or view the other users' preferences [19,20], while individual preferences are acquired when the emotions of the users are considered [12,13], or when a system combines the preferred music generes [36].

2.3 Modeling the preferences

Group modeling [33] combines multiple user models into a group model. Here, the modeling strategies employed in the literature are presented. An example of how each strategy works (which can be used as a reference during the description of each strategy) is first given, then each strategy is presented.

Group Modeling: Working Examples. Here, an example of how individual ratings are combined by each group modeling strategy is presented. In the example, three users (u_1, u_2, and u_3) rate ten items (identified by i_1, ..., i_{10}) with a value from 1 to 10. Table 1 reports the output of the strategies that combine individual ratings, while tables 2, 3, and 4, show how the *Borda Count*, *Copeland Rule*, and *Plurality Voting* strategies respectively work (the examples in these tables are based on the ratings in Table 1).

Additive Utilitarian [AU]. Individual ratings for each item are summed and a list of items ranked by sum is created. The produced list is the same when averaging the individual ratings, so this strategy is also called 'Average strategy'. An example of how it works is given in Table 1 (*AU* line).

The strategy has proven to be effective in different contexts [43], like the combination of preferences on different types of features (e.g., location, cost, cuisine) when recommending restaurants to a group [35].

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
u_1	8	10	7	10	9	8	10	6	3	6
u_2	7	10	6	9	8	10	9	4	4	7
u_3	5	1	8	6	9	10	3	5	7	10
$-AU$	20	21	21	25	26	28	22	15	14	23
$-MU$	280	100	336	540	648	800	270	120	84	420
$-AV$	2	2	3	3	3	3	2	1	1	3
$-LM$	5	1	6	6	8	8	3	4	3	6
$-MP$	8	10	8	10	9	10	10	6	7	10
$-AWM$	20	-	21	25	26	28	-	15	-	23
$-MRP$	8	10	7	10	9	8	10	6	3	6

Table 1 Output of the strategies that combine the original ratings

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
u_1	4.5	8	3	8	6	4.5	8	1.5	0	1.5
u_2	3.5	7.5	2	6.5	5	7.5	6.5	0.5	0.5	3.5
u_3	2.5	0	5	3	6	7.5	1	2.5	4	7.5
$-BC$	10.5	15.5	10	17	17	19.5	15.5	4.5	4.5	12.5

Table 2 Example of how the *Borda Count* strategy works, based on the ratings in Table 1

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
i_1	0	+	-	+	+	+	+	-	-	0
i_2	-	0	-	0	-	0	0	-	-	-
i_3	+	+	0	+	+	+	+	-	-	+
i_4	-	0	-	0	-	+	-	-	-	-
i_5	-	+	-	+	0	+	+	-	-	-
i_6	-	0	-	-	-	0	-	-	-	-
i_7	-	0	-	+	-	+	0	-	-	-
i_8	+	+	+	+	+	+	+	0	0	+
i_9	+	+	+	+	+	+	+	0	0	+
i_{10}	0	+	+	+	+	+	+	-	-	0
Index	-2	+6	-3	+6	+1	+8	+4	-8	-8	-2

Table 3 Example of how the *Copeland Rule* strategy works, based on the ratings in Table 1

	1	2	3	4	5	6
u_1	i_2, i_4, i_7	i_4, i_7	i_5	i_1	i_3	i_8
u_2	i_2, i_6	i_4, i_7	i_5	i_1, i_{10}	i_3	i_8, i_9
u_3	i_6, i_{10}	i_{10}	i_{10}	i_{10}	i_3	i_9
Group	i_2, i_6	i_4, i_7	i_5	i_1, i_{10}	i_3	i_8, i_9

Table 4 Example of how the *Plurality Voting* strategy works, based on the ratings in Table 1

Multiplicative Utilitarian [MU]. The ratings given by the users for each item are multiplied and a ranked list of items is produced. An example of how the strategy works is given in Table 1 (*MU* line).

This strategy was adopted by [14] in the music recommendation domain.

Borda Count [BC]. Each item gets a number of points, according to the position in the list of each user. The least favorite item gets 0 points and a point is added each time the next item in the list is considered. If a user gave the same rating to more items, points are distributed. So, in the example in Table 2, items i_8 and i_9 were rated by user u_2 with the lowest rating and share the lowest positions with 0 and 1 points, so both get (0+1)/2=0.5 points. A group preference is obtained by adding the individual points of an item.

This strategy was implemented in [4].

Copeland Rule [CR]. It is a form of majority voting that sorts the items according to their *Copeland index*, which is calculated as the number of times in which an alternative beats the others, minus the number of times it loses against the other alternatives. In the example in Table 3, item i_2 beats item i_1, since both users u_1 and u_2 gave it a higher rating.

The approach proposed in [17] proved that a form of majority voting is the most successful in a *requirements negotiation* context.

Plurality Voting [PV]. Each user votes for her/his favorite option. The one that receives the highest number of votes wins. If more than one alternative needs to be selected, the options that received the highest number of votes are selected. An example of how the strategy works is given in Table 4.

This strategy was implemented and tested by [48, 49] in the TV domain.

Approval Voting [AV]. Each user can vote for as many items as she/he wants, and a point is assigned to all the items a user likes. To show how the strategy works, in the example in Table 1 (*AV* line) we are going to suppose that each user votes for all the items with a rating above a threshold (for example, 5). A group preference is obtained by adding the individual points of an item.

To choose the pages to recommend to a group, *Let's Browse* [24] evaluates if the page currently considered by the system matches with the user profile above a certain threshold and recommends the one with the highest score. This strategy also proved to be successful in contexts in which the similarity between the users in a group is high [9].

Least Misery [LM]. The rating assigned to an item for a group is the lowest rating expressed for that item by any of the users in the group. This strategy usually models small groups, to make sure that every member is satisfied. A drawback is that if the majority of the group really likes something, but one person does not, the item will not to be recommended to the group. This is what happens in Table 1 for the items i_2 and i_7. An example of how the strategy works is given in Table 1 (*LM* line).

This strategy is used by [41], to recommend movies to small groups.

Most Pleasure [MP]. The rating assigned to an item for a group is the highest rating expressed for that item by a member of a group. An example of how the strategy works is given in Table 1 (*MP* line).

This strategy is used by [46] in a system that faces the cold start problem.

Average Without Misery [AWM]. The rating assigned to an item for a group is the average of the ratings given by each user. All the items that received a rating under a certain threshold by a user are not included in the group model (in the example in Table 1 - AWM line, the threshold rating is 4).

In order to model a group to decide the music to play in a gym, in [36] the individual ratings are summed, discarding the ones under a minimum degree.

Fairness [F]. This strategy is based on the idea that users can be recommended something they do not like, as long as they also get recommended something they like. This is done by allowing each user to choose her/his favorite item. If two items have the same rating, the choice is based on the other users' preferences. This is done until everyone made a choice. Next, everyone chooses a second item, starting from the person who chose last the first time.

If in the example in Table 1, we suppose that user u_1 chose first, so she/he would consider i_2, i_4, and i_7, and would choose i_4, because it has the highest average considering the other users' ratings. Next, u_2 would choose between i_2 and i_6 and would select i_6 for the same reason. Then, u_3 would choose item i_{10}. Since everyone chose an item, it would be u_3's turn again and i_5 would be chosen. User u_2 would choose i_2, which has the highest rating along with i_6 (which was already chosen). Then, u_1 would choose i_1, which is the one with the highest rating and was not chosen yet. The final sequence of items that models the group would be: i_4, i_6, i_{10}, i_5, i_2, i_1, i_3, i_8, i_9.

This strategy is adopted by [14] in the music recommendation context.

Most Respected Person (Dictatorship) [MRP]. This strategy selects the items according to the preferences of the most respected person, using the preferences of the others just in case more than an item received the same evaluation. The idea is that there are scenarios is which a group is guided/dominated by a person. In the example in Table 1, u_1 is the most respected person.

This strategy is adopted to select tourist attractions advantaging the users with particular needs [2], or when experts are recognized in a group [22]. Moreover, there are studies that highlight that when people interact, a user or a small portion of the group influences the choices of the others [10].

2.4 Rating Prediction

Group preferences can be generated by using one of the following three approaches [21]: (a) generation of a group model, which combines individual preferences and is used to build predictions for the group, (b) merging of recommendations built for individual users, or (c) aggregation of the predictions built for every user.

2.4.1 Construction of Group Preference Models

This approach builds a group model by using the preferences expressed by each user, then predicts a rating for the items that do not have a score in the group model. The approach follows two main steps, described below.

1. Construct a model M_g for a group g, which contains its preferences
2. For each item i not rated by the group, use M_g to predict a rating p_{gi}

Group predictions have been employed for people browsing the web together [24], to select the music to play for people working out [36], or to consider the preferences of subgroups with similar needs in the tourist domain [2].

2.4.2 Merging of Recommendations Made for Individuals

This approach presents to a group a set of items, which is the merging of the items with the highest predicted rating for each user in the group. The approach works as follows.

1. For each member of the group u:
   – For each item i not rated, predict a rating p_{ui}
   – Select the set C_i of items with the highest predicted ratings p_{ui}
2. Model the preferences of each group by producing $\bigcup_i C_i$, i.e., the union of the sets of items with the highest predicted ratings of each member

This approach is implemented by *PolyLens* [41] and adopted by [4].

2.4.3 Aggregation of Individual Predictions

This approach first predicts individual preferences for all the items not rated by each user, then aggregates individual preferences for an item, in order to derive a group preference. The approach works as follows:

1. For each item i:
   – For each user u of the group g that did not rate i, predict a rating p_{ui}
   – Calculate an aggregate rating r_{gi} from the ratings of the users in the group, either expressed (r_{ui}) or predicted (p_{ui})

This approach is adopted by *PolyLens* [41] that, in order to recommend movies to a group, first predicts individual ratings and then aggregates them.

2.5 Explanation of the recommendations

Since group recommender systems deal with the preferences of more than a user, an aspect considered by some systems is the explanation of the recommendation, i.e., why the proposed items have been selected for the group.

In the literature, explanations are used to justify why a page has been recommended to a group of users by presenting the commonalities between the users [24], why a tourist attraction has been recommended by displaying the features that characterized its selection [2], or what music is likely to be played next according to the song currently being played [15].

2.6 Help the members to achieve consensus

Once the preferences of a group have been modeled and group predictions have been produced, the list of items to recommend to the group has to be selected. In order to do so, three strategies are usually employed by the systems:

1. the system suggests the items with the highest predicted ratings, without consulting the group;
2. a member of the group is responsible for the final decision;
3. the users in the group have a conversation, in order to achieve consensus.

The vast majority of the systems automatically selects the items to recommend. An example of a system that bases the selection of the items on the decision of a member is [2], which suggests the attractions for a group, but expects the tour guide to choose the ones that the group should visit. Regarding the systems that employ a conversation among the group, in [37,38] the users reach a consensus by interacting through a shared device.

2.7 Motivation and Contribution

This section presented group recommendation, both from a design and a literature point of view. An important aspect to notice is that no approach in the literature considers a scenario in which the number of recommendation lists is limited. Even if in [39,40] clustering is used in a group recommendation approach, similarly to what we have done in previous works [6–8], clusters are used only to select the number of neighbors of a user when predictions are built, in order to speed up the computation. This is the first time that the group recommendation with automatic group detection is explored in its entirety, and the next sections will present the work on each component of a system.

3 Problem Statement

As highlighted in the previous section, the approaches in the literature handle groups that are already formed, and a group recommender system that also detects groups does not exist. Before presenting a formalization of the problem handled by *ART*, we introduce the problem of producing recommendations for individual users. This definition mainly follows the one proposed in [16].

Let $U = \{u_1, u_2, ..., u_n\}$ be a set of users, $I = \{i_1, i_2, ..., i_m\}$ be a set of items, and V be a totally ordered set of values that can be used to express a preference. The set of ratings expressed in the system is a ternary relation

$R \subseteq U \times I \times V$; each rating is denoted by r_{ui}. The subset of users who rated an item i is denoted by U_i and the subset of items rated by a user u can be denoted by I_u. All the items in I that have been rated by both a user u and a user v can be indicated by $I_{uv} = I_u \cap I_v$, and all the users in U that rated both an item i and an item j can be denoted by $U_{ij} = U_i \cap U_j$.

The main recommendation problem consists in the identification for a user $u \in U$ of an item $i \in I \backslash I_u$ that user u is likely to be interested in. The task is a regression or multi-class classification problem, in which the goal is the learning of a function $f : U \times I \rightarrow V$ that predicts a rating $p_{ui} = f(u, i)$ for a user u of an item i. The objective is the identification of an item i^* for the user u, which has the highest estimated rating:

$$i^* = \operatorname*{argmax}_{j \in I \backslash I_u} f(u, j). \tag{1}$$

The generation of group recommendations with the automatic detection of groups can be formalized starting from the previous problem, by adding one more constraint. Given a value k, which denotes the maximum number of recommendation lists that can be generated, the additional requirement is to split the set of users U in a partition of k groups, so that for each group $G_q \subseteq U$ $(q \in \{1, ..., k\})$ every user $u \in G_q$ receives the same recommendations. Moreover, $\forall u \in U$ $\exists!$ $q \in \{1, ..., k\}$ s.t. $u \in G_q$. So, each user has to belong to a group in order to receive a recommendation, and given two groups G_q and G_w, $\forall q, w \in \{1, ..., k\}$ $q \neq w \Rightarrow G_q \cap G_w = \emptyset$ (i.e., groups are formed by subsets of users that do not intersect, in order for a user to belong to just one group and receive only a list of recommended items). The goal is the learning of a function $f : G_q \times I \rightarrow V$ that predicts a rating $p_{qi} = f(G_q, i)$ for a group G_q of an item i. In this case, the objective is the identification of an item i^* for the active group G_q, that has the highest estimated rating:

$$i^* = \operatorname*{argmax}_{j \in I} f(G_q, j). \tag{2}$$

Unlike Equation 1, in which the set of items that can be recommended is the set of items that were not evaluated by a user $(I \backslash I_u)$, in Equation 2 the items that can be recommended to a group can be taken from whole set I. This is because with large groups there might not be items that were not evaluated by anyone and the system might not produce recommendations to the group. This intuition is confirmed by analyzing the datasets employed to validate our proposal. Indeed, if in each clustering created in the evaluation presented in Section 5 we avoid recommending the items that have already been evaluated by at least a user, between 5% and 10% of the groups would have an empty recommendation list. Thinking about a real-world group recommendation campaign like the one presented in the Introduction, this would not be acceptable and would create significant losses for a company. In our proposal, we will deal with the already evaluated items in different ways.

	Step 1	Step 2	Step 3
1. ModelBased	Clustering	Group Modeling	Group Predictions
2. MergeRecommendations	Clustering	Individual Predictions	Group Modeling (*top-n* items)
3. PredictionsAggregation:Cluster&Predict	Clustering	Individual Predictions	Group Modeling (all items)
4. PredictionsAggregation:Predict&Cluster	Individual Predictions	Clustering	Group Modeling (all items)

Table 5 Classes of systems that compose *ART*

Clustering	Group Modeling	Group Predictions
k-means	Additive Utilitarian	Item-based Collaborative Filtering
k-means	Borda Count	Item-based Collaborative Filtering
k-means	Approval Voting	Item-based Collaborative Filtering
k-means	Least Misery	Item-based Collaborative Filtering
k-means	Most Pleasure	Item-based Collaborative Filtering
k-means	Average Without Misery	Item-based Collaborative Filtering

Table 6 Systems that compose the *ModelBased* class

4 Algorithms

This section presents *ART (Automatic Recommendation Technologies)*, a set of group recommender systems able to produce suggestions respecting a constraint on the number of recommendation lists that can be generated.

The proposed systems combine a set of algorithms that complete a different task of a group recommender system. Each algorithm is based on a state-of-the-art solution, which has been adapted to improve the accuracy of the proposed group recommender systems.

Table 5 summarizes how each class of systems works.

4.1 Class of Systems Based on Group Model Construction

ModelBased is a class of systems that detect groups of similar users, model each group by using the preferences of its members, and predict group preferences by using the model, according to the approach presented in 2.4.1.

Here, we describe the steps performed by the systems that belong to this class and how they have been implemented. Each system works in three steps:

1. *Detection of the groups (Clustering).* Groups of users with similar preferences are detected through the k-means clustering algorithm [25].
2. *Group modeling.* Once groups have been detected, a group model is built for each group, by using one of the modeling strategies presented in 2.3. Each system implements one of the six modeling strategies that could be applied to our context in which groups have to detected. For each group, a rating is calculated for a subset of items.
3. *Prediction of the Missing Ratings Using a Group Model.* A group rating is predicted for each item not modeled by the previous step, by using the model that contains its preferences with an adapted version of the Item-based Collaborative Filtering approach presented in [47].

The systems belong to the first class illustrated in Table 5. For each system of the *ModelBased* class, Table 6 summarizes with which algorithms each step has been implemented. All the steps are now described in detail.

Detection of the groups (Clustering). In order to respect the constraint imposed by the context, the set of users has to be partitioned into a number of groups equal to the number of recommendation lists that can be produced. Since groups do not exist, unsupervised classification (*clustering*) is necessary. In [1], it was highlighted that the k-means clustering algorithm is the most used in recommender systems. Moreover, we previously analyzed [7] and compared [8] a different option to group the users, by using the Louvain community detection algorithm, which produces a hierarchical partitioning of the users; however, results showed that k-means is more accurate, so this approach is also used in this study. The input of the algorithm is a rating matrix that contains a row for each user, a column for each item, and whose elements are the ratings given by each user for the corresponding item. The distance between the users is computed with the Euclidean distance, and the output is a partitioning of the users into groups (clusters), such that users with similar models (i.e., similar ratings for the same items) are in the same group and receive the same recommendations.

Group modeling. In order to create a model that represents the preferences of a group, a subset of the group modeling strategies previously described is employed. We considered all the ones presented in [33] (previously described in 2.3), and implemented an adapted version of the strategies that could be applied to our context in which groups have to be detected. In fact, there are some strategies that do not produce a rating, but just a ranked list of the items (i.e., the *Plurality Voting*, *Copeland Rule*, and *Fairness* strategies), and there is a strategy (i.e., *Most Respected Person*) where just the ratings of the user who guides the group are considered (a "most respected person" does not exist in an automatically detected group). The *Multiplicative Utilitarian* strategy could not be tested because of the limit on the maximum number that can be calculated by a computer[1]; normalizing the ratings led to a loss in precision, and to a consequent drop in the accuracy of the system.

The other strategies, i.e., *Additive Utilitarian*, *Borda Count*, *Approval Voting*, *Least Misery*, *Most Pleasure*, and *Average Without Misery*, were employed.

As it can be noticed from tables 1, 2, and 3, the produced group ratings are in different ranges (i.e., individual ratings are between 1 and 10, while group ratings can be much higher than 10). In order to evaluate how each group rating reflects the individual preferences, it is necessary for both individual and group ratings to be in the same range. This can be obtained with a simple proportion:

$$group\_rating : max\_group\_rating = new\_group\_rating : max\_user\_rating \tag{3}$$

where:

– *group_rating* is the rating produced by a modeling strategy;

[1] A 64 bits machine cannot calculate numbers higher than 2^{52}. Even considering only 55 users that all gave a very small rating for an item, like 2, an overflow would occur ($2^{55} > 2^{52}$).

- *max_group_rating* is the maximum value of *group_rating* that can be obtained for an item;
- *max_user_rating* is the maximum rating that a user can give to an item.

So, considering the proportion in Equation 3, *new_group_rating* can be derived as follows:

$$new\_group\_rating = \frac{group\_rating \cdot max\_user\_rating}{max\_group\_rating} \quad (4)$$

For some of the strategies, the proportion has been adapted as follows:

- Considering the ratings produced by the *Approval Voting* strategy, the value of *max_group_rating* does not take a fixed value in the whole dataset, but it needs to be evaluated for each group that is being modeled. In fact, *max_group_rating* is equal to the number of users in a group, since an item might receive a point by all the users in a group.
- The peculiarity of the *Borda Count* strategy is that *max_group_rating* cannot take the same value for all the items, since the strategy creates a ranking of each user's preferences. Therefore, a mapping of the range of values created by the strategy to the range of values of the individual ratings is necessary. The values considered in the mapping are:
  - *group_rating*, which is the rating produced by the modeling strategy for an item;
  - *min_group_rating*, which is the minimum value of *group_rating* that can be obtained for an item;
  - *max_group_rating*, which is the maximum value of *group_rating* that can be obtained for an item;
  - *min_user_rating*, which is the minimum rating that a user can give to an item;
  - *max_user_rating*, which is the maximum rating that a user can give to an item.

  In order to make the mapping, we first calculate how wide each range is:

$$modeled\_ratings\_span = max\_group\_rating - min\_group\_rating \quad (5)$$

$$users\_ratings\_span = max\_user\_rating - min\_user\_rating \quad (6)$$

  Then, we convert the range produced by the strategy into a 0-1 range:

$$modeled\_value = (group\_rating - min\_group\_rating)/modeled\_ratings\_span \quad (7)$$

  Then, the 0-1 range is converted into the desired range:

$$new\_group\_rating = min\_user\_rating + (modeled\_value \cdot user\_ratings\_span) \quad (8)$$

An aggregate rating r_{gi} can be considered as significant for a group g and included in the model only if a consistent part of the group has rated item i. In fact, if an item is rated by a small part of the group, the aggregate rating cannot be considered as representative of the group as a whole. So, a parameter named *coratings*, which indicates the minimum percentage of the group who has to rate an item in order to include the rating in the model, is set.

Prediction of the Missing Ratings Using a Group Model. In the group models created by the previous step, for a subset of items there is no preference. In order to estimate these preferences, a rating p_{gi} for an item i not rated by a group g is predicted through the model that contains the group preferences. This is done by using an Item-Based Nearest Neighbor Collaborative Filtering algorithm [47]. This choice was made since for large groups it would not be significant to look for "similar groups" with a User-based approach[2].

The algorithm predicts a rating p_{gi} for each item i that was not evaluated by a group g, by considering the rating r_{gj} of each similar item j rated by the group. Equation 9 gives the formula used to predict the ratings:

$$p_{gi} = \frac{\sum_{j \in ratedItems(g)} itemSim(i,j) \cdot r_{gj}}{\sum_{j \in ratedItems(g)} itemSim(i,j)} \tag{9}$$

Some authors do not consider all the items rated in the model (i.e., the set $ratedItems(g)$), but just the top n correlations [47]. In order to reduce the complexity of the algorithm and select only meaningful correlations, this approach is used for this step. To indicate that we are dealing with the most similar items, the top n correlations will be indicated as $neighbors^{ii}$.

In order to compute similarity $itemSim(i,j)$ between two items, adjusted-cosine similarity is used. This metric is the most accurate to calculate item similarity [47]. Equation 10 gives the formula to compute it (as mentioned in Section 3, U_{ij} is the set of users that rated both item i and j); note that in this formula $U_{ij} \subseteq g$, i.e., only the ratings of the group are considered.

$$itemSim(i,j) = \frac{\sum_{u \subset U_{ij}} (r_{ui} - \overline{r}_u)(r_{uj} - \overline{r}_u)}{\sqrt{\sum_{u \subset U_{ij}} (r_{ui} - \overline{r}_u)^2} \sqrt{\sum_{u \subset U_{ij}} (r_{uj} - \overline{r}_u)^2}} \tag{10}$$

Value $\overline{r}_u$ represents the mean of the ratings given by user u.

4.2 System that Merges Individual Recommendations

MergeRecommendations is a system that detects groups of similar users, predicts individual preferences, and selects the items with the highest predicted ratings for each user, using the approach presented in 2.4.2.

Here, we describe the steps performed by the system and how they have been implemented. The system works in three steps:

1. *Detection of the groups (Clustering).* Considering the user models that contain individual preferences, groups of similar users are detected through the k-means clustering algorithm.

[2] Considering an example with 6000 users and 10 groups of homogeneous size, if a User-Based approach was used, when looking for neighbors the algorithm would look for two similar models that each contain a synthesis of the preferences of 600 users, and this type of similarity would not be accurate enough to make the predictions.

Clustering	Individual Predictions	Group Modeling (*top-n* items)
k-means	User-based Collaborative Filtering	Additive Utilitarian

Table 7 System that composes the *MergeRecommendations* class

2. *Prediction of the missing ratings.* Individual predictions are built for each user with a User-Based Collaborative Filtering Approach presented in [47].
3. *Generation of the group predictions (Group modeling).* Group predictions are built by modeling the *top-n* items with the highest predicted ratings for each user (i.e., the individual recommendations produced for each user), by averaging the ratings of the items selected for each user.

The system belongs to the second class illustrated in Table 5. Table 7 summarizes with which algorithms each step has been implemented. Note that *MergeRecommendations* is composed by a single system, since the group modeling step has been implemented only with a strategy. The choice to model the ratings of the items that appear in more than a list with an average (*Additive Utilitarian*) and not with other strategies, was made considering that the approach that aggregates the predictions is a union of the individual lists. Therefore, all the users have the same importance and no rating takes a different weight in the group model. All the steps are now described in detail.

Detection of the Groups (Clustering). The first step uses the same approach previously presented for the *ModelBased* class of systems, i.e., k-means.

Prediction of the Missing Ratings. The ratings for the users in a group are predicted with a widely-used User-Based Nearest Neighbor Collaborative Filtering algorithm, presented in [47]. The algorithm predicts a rating p_{ui} for each item i that was not evaluated by a user u, by considering the rating r_{ni} of each similar user n for the item i. A user n similar to u is called a *neighbor* of u[3]. Equation 11 gives the formula used to predict the ratings:

$$p_{ui} = \overline{r}_u + \frac{\sum_{n \subset neighbors^{uu}(u)} userSim(u,n) \cdot (r_{ni} - \overline{r}_n)}{\sum_{n \subset neighbors^{uu}(u)} userSim(u,n)} \tag{11}$$

Values $\overline{r}_u$ and $\overline{r}_n$ represent, respectively, the mean of the ratings expressed by user u and user n. Similarity $userSim()$ between two users is calculated using the Pearson's correlation, a coefficient that compares the ratings of all the items rated by both the target user and the neighbor. Pearson's correlation between a user u and a neighbor n is given in Equation 12. As mentioned in the Introduction, I_{un} is the set of items rated by both user u and user n.

$$userSim(u,n) = \frac{\sum_{i \subset I_{un}} (r_{ui} - \overline{r}_u)(r_{ni} - \overline{r}_n)}{\sqrt{\sum_{i \subset I_{un}} (r_{ui} - \overline{r}_u)^2} \sqrt{\sum_{i \subset I_{un}} (r_{ni} - \overline{r}_n)^2}} \tag{12}$$

[3] To indicate that we are dealing with a user-based approach, the set of neighbors of this algorithm will be indicated as $neighbors^{uu}$.

Clustering	Individual Predictions	Group Modeling (all items)
k-means	User-based Collaborative Filtering	Additive Utilitarian
k-means	User-based Collaborative Filtering	Borda Count
k-means	User-based Collaborative Filtering	Approval Voting
k-means	User-based Collaborative Filtering	Least Misery
k-means	User-based Collaborative Filtering	Most Pleasure
k-means	User-based Collaborative Filtering	Average Without Misery

Table 8 System that composes the *PredictionsAggregation:Cluster&Predict* class

The values of the metric range from 1.0 (complete similarity) to -1.0 (complete dissimilarity). Negative correlations do not increase the prediction accuracy [18], so they are discarded. An adapted version of this algorithm that considers only the users in the group to build the predictions was implemented, but it was less accurate than the one that uses the whole user set.

Generation of the Group Predictions (Group modeling). For each user, the items for which a rating is predicted are ranked in descending order based on the ratings, then the *top-n* items are selected. Group ratings are predicted by modeling the *top-n* items with a union of the items that have the highest predicted ratings for each user. If an item appears in the list of more members of the same group, the average of the ratings for that item is calculated.

4.3 Classes of Systems Based on the Aggregation of Individual Predictions

Here, we present two classes of systems that aggregate individual predictions, which perform the same steps in different order to improve their accuracy.

4.3.1 PredictionsAggregation:Cluster&Predict

This class of systems detects groups of similar users, predicts individual preferences, and aggregates the preferences expressed for each item into a group preference, according to the approach presented in 2.4.3.

Here, we describe the steps performed by the systems that belong to this class and how they are implemented. The systems work in three steps:

1. *Detection of the groups (Clustering).* Using a set of user models that contain individual preferences, groups of users with similar preferences are detected through the k-means clustering algorithm.
2. *Prediction of the missing ratings.* Individual predictions are built for each user with the User-Based Collaborative Filtering approach.
3. *Aggregation of the Individual Preferences (Group modeling).* Groups are modeled by using one of the six previously employed strategies.

The systems belong to the third class illustrated in Table 5. For each system that belongs to the *PredictionsAggregation:Cluster&Predict* class, Table 8 summarizes with which algorithms each step has been implemented.

The first two steps use the same algorithms previously presented, i.e., k-means and the User-Based Collaborative Filtering algorithm. Therefore, only the approach used to aggregate individual preferences is now described.

Aggregation of the Individual Preferences (Group modeling). The same modeling strategies used for *ModelBased* can be employed by the systems. However, these systems model individual predictions, which have been calculated for all the items not rated by a user. Therefore, there is no need to remove any group prediction from the model by introducing a *coratings* parameter.

4.3.2 PredictionsAggregation:Predict&Cluster

The class of systems *PredictionsAggregation:Cluster&Predict* previously presented detects groups of users by using the preferences of the users for the evaluated items. However, the number of items rated by the users is much lower than the number of available items. Sparsity leads to the *curse of dimensionality* phenomena [5] that are known to affect the clustering algorithms.

PredictionsAggregation:Predict&Cluster was conceived to improve the quality of the clustering of *PredictionsAggregation:Cluster&Predict*. *PredictionsAggregation:Predict&Cluster* detects groups by giving as input to k-means not only the ratings explicitly expressed by users, but also the predicted values of the unrated items. In order to do so, the individual predictions are calculated by the systems in *PredictionsAggregation:Predict&Cluster* at the beginning of the computation. By using more values as input for the clustering, the algorithm should identify groups of users with more correlated preferences. This novel approach to user clustering allows a system to remove sparsity and overcome the curse of dimensionality phenomena, without increasing the complexity of the system.

In conclusion, *PredictionsAggregation:Predict&Cluster* performs the same steps of the systems in *PredictionsAggregation:Cluster&Predict* but computes the predictions before clustering the users. The preferences expressed by the users and the individual predictions are also used to model the groups.

5 Experimental Framework

The evaluation of the proposed systems involved hundreds of experiments and statistical tests, and this section presents the framework used to perform them.

5.1 Strategy

In order to perform the experiments, we adopted MovieLens-1M and Yahoo! Webscope (R4), which are two of the most widely adopted datasets in the recommender systems literature. In the literature it has been highlighted that no dataset that contains a partitioning of the users into groups exists [42, 46]. The existing group recommender system either add constraints on a dataset like the ones we adopted (the vast majority), obtain datasets from social networks that involve the concept of group like Flickr [50] and Facebook [3], or perform use-case studies to perform the evaluation [37]. This analysis shows that no public dataset contains information about the partitioning of the users into groups and of their preferences. Therefore, comparing our proposal with a case in which groups were manually created instead of being automatically detected, would not allow us to have a real comparison with the types of groups handled in the literature (e.g., an established and consolidated group of users). Our choice has been to compare our proposal with two more realistic scenarios, by (i) comparing the accuracy of each group recommender system that automatically detects groups with that of a system that produces individual recommendations, and (ii) by comparing the accuracy of our systems that automatically detect groups with a case in which there is no level of personalization and all the users receive the same recommendations (more details of this comparisons will be presented in a few paragraphs).

The clusterings with k-means were created by the KMlocal [23] program, which contains a variant of the k-means algorithm, called *EZ Hybrid*. K-means minimizes the *average distortion*, i.e., the mean squared Euclidean distance from each data point to its nearest center. We performed the experiments with all the algorithms contained in the KMlocal program (i.e., *Lloyd's*, *Swap*, *Hybrid*, and *EZ Hybrid*), and in all the four clusterings and all the classes of systems *EZ Hybrid* is the algorithm that returned the lowest distortion[4], so it is the one used to cluster the users.

In order to identify the most performing system, we compared the RMSE values obtained by each of them, considering different numbers of groups to detect (i.e., the number of recommendation lists that can be produced by the system), and different values for a system's parameters.

In order to evaluate if two RMSE values of two systems are significantly different, independent-samples two-tailed Student's t-tests have been performed, so each experiment was repeated five times with a 5-fold cross-validation.

The details of the experiments are described below.

1. *Selection of the best system.* For each class, the most accurate system is chosen. In order to evaluate the prediction accuracy for different numbers of groups, i.e., for different numbers of recommendation lists that can be produced, four clusterings in 20, 50, 200, and 500 groups were created.
   Moreover, we compared the results obtained with the previously mentioned four clusterings with the results obtained by considering a single group with all the users (i.e., we tested the systems in a scenario in which just one recommendation list can be produced, so predictions for an item are calculated considering the preferences of all the users), and the results obtained by the system that calculates individual predictions for each user (i.e., we simulated the case where there is no constraint, in order to compare the performance of the systems when they work with groups)[5].
2. *Setting of the parameters.* Considering the most performing system of each of the four classes, a parametric analysis has been performed, in order to identify the setting that allows to achieve the best accuracy for the system.

[4] The details of these experiments are omitted to facilitate the readability of the paper.
[5] In the figures, the results of these two settings will be joined with a dashed line.

3. *Selection of the best class.* The four best settings have been compared as the number of groups changes, in order to identify the most accurate system.

5.2 Datasets and Data Preprocessing

The two datasets adopted to validate the proposed systems are MovieLens-1M[6] and Yahoo! Webscope (R4)[7]. Here, we describe the details of the datasets and the preprocessing performed on them.

5.2.1 MovieLens-1M

The first dataset used to perform the experiments, i.e., MovieLens-1M, is composed of 1 million ratings, expressed by 6040 users for 3900 movies. The dataset sparsity is 95.53%.

For this framework, only the file `ratings.dat` that contains the actual ratings given by the users is considered. The file contains four features: *UserID*, which contains the user IDs in a range between 1 and 6040, *MovieID*, which contains the IDs of the movies in a range between 0 and 3952, *Rating*, which contains values in a scale between 1 and 5, and *Timestamp*, which contains a timestamp of the moment in which a user rated an item. The file was preprocessed for the experimentation. Out of all the features, just the first three were selected (i.e., *UserID*, *MovieID*, and *Rating*), since none of the presented algorithms uses a timestamp.

5.2.2 Yahoo! Webscope (R4)

The second dataset adopted in our experiments, i.e.,Yahoo! Webscope (R4), contains a large amount of data related to users preferences expressed by the Yahoo! Movies community that are rated on the base of two different scales, from 1 to 13 and from 1 to 5 (we use the latter). The dataset contains 211197 ratings, given by 7642 users to evaluate 11915 items. The dataset sparsity is 98.16%.

Given the high number of items involved in the dataset, which represent the features in the clustering, and the high sparsity that characterizes the data, a sample was extracted, by removing all the users who evaluated less that 17 items and all the items that have been evaluated by less than 13 users[8]. The final sample consists of 5070 users, 1647 items, and 153461 ratings.

[6] http://www.grouplens.org/

[7] http://www.cs.umd.edu/~mount/pubs.html

[8] These values have been chosen in order to have a dataset in which useful information about each user and each item was available to make the predictions.

5.2.3 Partitioning into Training and Test Sets

The experiments were repeated five times with a 5-fold cross-validation, in order to perform statistical tests to validate the results. Therefore, each dataset was split in five subsets with a random sampling technique. To ensure that every user had ratings both in the training and the test sets, every subset contains a number of ratings for each user proportional to the number of ratings she/he gave.

5.3 Metrics

The accuracy of the predicted ratings was measured through the Root Mean Squared Error (RMSE). The metric compares the test set with the predicted ratings, by comparing each rating r_{ui}, given by a user u for an item i, with the rating p_{gi}, predicted for the item i for the group g in which user u is. The formula is shown below (n is the number of ratings available in the test set):

$$RMSE = \sqrt{\frac{\sum_{i=0}^{n} (r_{ui} - p_{gi})^2}{n}}$$

In order to compare if two RMSE values returned by two experiments are significantly different, independent-samples two-tailed Student's t-tests have been performed. These tests allow us to reject the null hypothesis that two values are statistically the same and the difference in the values is due to chance. So, a two-tailed test will evaluate if a RMSE value is significantly greater or significantly smaller than another RMSE value. Since each experiment was performed five times, the means M_i and M_j of the RMSE values obtained by two algorithms i and j are used to compare them and calculate a value t:

$$t = \frac{M_i - M_j}{s_{M_i - M_j}}$$

where

$$s_{M_i - M_j} = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$$

s^2 is the variance of the two samples, n_1 and n_2 are the number of values used to build M_1 and M_2 (in our case both are equal to 5, since the experiments were repeated five times). In order to derive the *t-value* that indicates the result of a test, the degrees of freedom for the test have to be determined:

$$\text{d.f.} = \frac{(s_1^2/n_1 + s_2^2/n_2)^2}{(s_1^2/n_1)^2/(n_1 - 1) + (s_2^2/n_2)^2/(n_2 - 1)}$$

Given t and $d.f.$, the *t-value* (i.e., the result of the test), can be obtained in a standard table of significance as

$$t(d.f.) = t\text{-}value$$

The *t-value* allows us to derive the probability p that there is no difference between the two means. In order to facilitate the reading of the paper, only the result of each test is presented (i.e., the probability p of two accuracy values to be different due to chance).

5.4 Experiments

The evaluation of each class of systems is now presented.

5.4.1 ModelBased - Selection of the Best System

In this experiment, the systems that use the different modeling strategies are tested. If a strategy uses a threshold (i.e., *Approval Voting* and *Average Without Misery*), all the possible values are tested. The other parameters are set with a fixed value, i.e., *coratings* is set to 10% and *neighbors*[ii] is set to 10.

Fig. 1 and Fig. 2 show the RMSE values for each system that uses a different modeling strategy and each clustering, respectively for the MovieLens-1M and the Webscope (R4) datasets. Note that the results obtained for the *Average Without Misery* strategy with all the threshold values and the *Approval Voting* strategy with threshold values 3 and 4 have been omitted, since a small portion of the test set could be considered (less than 10%) and results were not reliable[9]. The details of the experiments of the three most performing strategies are zoomed in Fig. 3 and Fig. 4 to improve their analysis, since the results are very close. As it can be noticed, *Additive Utilitarian* and *Average Without Misery* trivially lead to the same accuracy, as removing the threshold from the second strategy leads to a group model that averages the individual ratings. Therefore, in the next experiments that employ a modeling strategy, the results of *Average Without Misery* will be discarded. In every clustering, *Additive Utilitarian* is the strategy that best models the groups. Since the system deals with large groups, this is why an average, which is a single value that is meant to typify a set of different values, is best way to combine the ratings in this context. This result is also strengthened by the fact that *Least Misery* (i.e., the strategy that assigns to the group the lowest value assigned by a user) is the strategy that performs worst. In fact, a large group is modeled just with low ratings, since the probability to have a user who did not like an item is high. The models produced by *Borda Count*, which preserves the ranking of the items but modifies their ratings, lead to a slightly lower accuracy with respect to *Additive Utilitarian*.

[9] This happens because *Average Without Misery* discards a group prediction if at least a user has evaluated an item with a rating lower than the threshold. Therefore, even with a small threshold, like 2, the vast majority of the items is not modeled by the strategy in a context in which groups are large (the larger is the group, the higher is the number of times an item is rated, and higher is the probability that at least one user did not like the item). Strategy *Approval Voting* with threshold values 3 and 4 could not be evaluated, because considering only items with a high rating (i.e., with a rating above 3), too many ratings are discarded from the model, and a low percentage of comparisons with the test set was done.

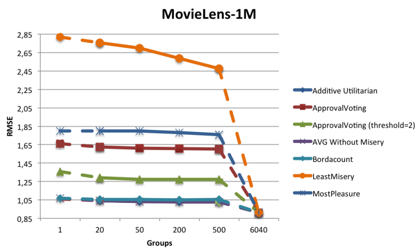

Fig. 1 RMSE values of *ModelBased* for the different modeling strategies in the MovieLens-1M dataset

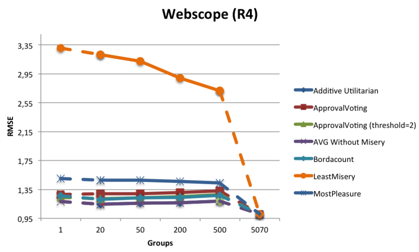

Fig. 2 RMSE values of *ModelBased* for the different modeling strategies in the Webscope (R4) dataset

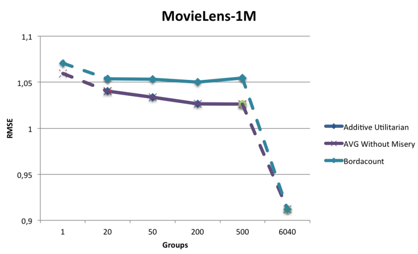

Fig. 3 Detail of the experiment to study the modeling strategies in the MovieLens-1M dataset

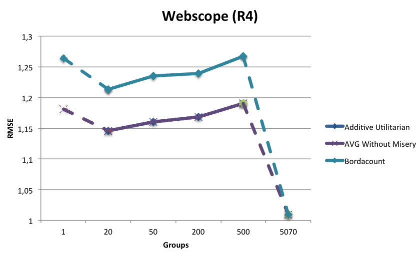

Fig. 4 Detail of the experiment to study the modeling strategies in the Webscope (R4) dataset

Regarding the difference between the results obtained by the two datasets, we can notice that in the Webscope (R4) dataset, the *Approval Voting* and *Most Pleasure* strategies have a lower gap with *Additive Utilitarian*, with respect to the behavior in the MovieLens-1M dataset. Moreover, we can notice that apparently in the Webscope (R4) dataset the results for 200 and 500 groups worsen. However, the results of independent samples Student's t-tests showed that the difference in the results is not significant (they have been omitted to facilitate the reading of the paper). This apparent worsening is the sign that a model-based group recommender system is not accurate for small groups. More specifically, given a small amount of information about the preferences for the items, a group model cannot be effectively employed to generate group recommendations.

Independent-samples t-tests, which compared the results for each pair of modeling strategies, returned that there is a significant difference in the values obtained with each modeling strategy. For readability reasons, just the comparison between *Additive Utilitarian* (*AU*) and *Borda Count* (*BC*) is reported in Table 9 and Table 10 (g is used to indicate the number of groups).

MovieLens-1M	$RMSE_{AU}$ **vs.** $RMSE_{BC}$
g=1	$p = 0.00$
g=20	$p = 0.00$
g=50	$p = 0.00$
g=200	$p = 0.00$
g=500	$p = 0.00$

Table 9 Results of the the independent-samples two-tailed Student's t-tests that compare the accuracy obtained by employing different group modeling strategies.

Webscope (R4)	$RMSE_{AU}$ **vs.** $RMSE_{BC}$
g=1	$p = 0.00$
g=20	$p = 0.00$
g=50	$p = 0.00$
g=200	$p = 0.00$
g=500	$p = 0.00$

Table 10 Results of the the independent-samples two-tailed Student's t-tests that compare the accuracy obtained by employing different group modeling strategies.

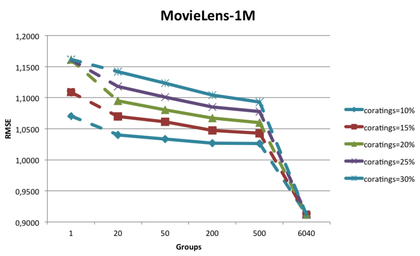

Fig. 5 RMSE values for the different values of the *coratings* parameter in the MovieLens-1M dataset

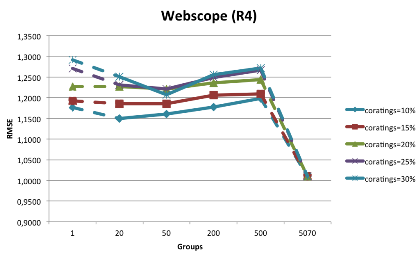

Fig. 6 RMSE values for the different values of the *coratings* parameter in the Webscope (R4) dataset

5.4.2 ModelBased - Setting of the Parameters

Here, a description of the experiments performed to set the two systems' parameters (i.e., *coratings* and $neighbors^{ii}$) is presented.

coratings parameter setting. The values of the parameter, which allows a system to consider in the model only the items rated by a certain part of the group, will be tested in the system that builds the models with *Additive Utilitarian*; parameter $neighbors^{ii}$ is set to 10. Fig. 5 and Fig. 6 clearly show that the initial value of *coratings*, i.e., 10%, is the one that allows the system to achieve the best results. This means that the higher is the value of *coratings*, the more ratings are eliminated from the model and the harder it is for the system to predict ratings for a group.

Independent-samples t-tests, performed to compare the results for different values of *coratings* in each clustering, returned that there is a significant difference in the RMSE obtained with different values of *coratings*. The tests that considered 10% and 15% of the group are reported in Table 11 and Table 12 (g is used to indicate the number of groups). The results suggest lowering the *coratings* value improves the results.

Setting of the parameter $neighbors^{ii}$*.* Parameter $neighbors^{ii}$ allows a system to select the number of similar items to consider when building a group prediction. The aspects previously tested are set as previously illustrated, i.e., the modeling strategy is *Additive Utilitarian* and *coratings* = 10%.

MovieLens-1M	$RMSE_{10\%}$ **vs.** $RMSE_{15\%}$
g=1	$p = 0.00$
g=20	$p = 0.00$
g=50	$p = 0.00$
g=200	$p = 0.00$
g=500	$p = 0.00$

Table 11 Results of the the independent-samples two-tailed Student's t-tests that compare the accuracy obtained by employing different values of the *coratings* parameter.

Webscope (R4)	$RMSE_{10\%}$ **vs.** $RMSE_{15\%}$
g=1	$p = 0.00$
g=20	$p = 0.00$
g=50	$p = 0.00$
g=200	$p = 0.00$
g=500	$p = 0.00$

Table 12 Results of the the independent-samples two-tailed Student's t-tests that compare the accuracy obtained by employing different values of the *coratings* parameter.

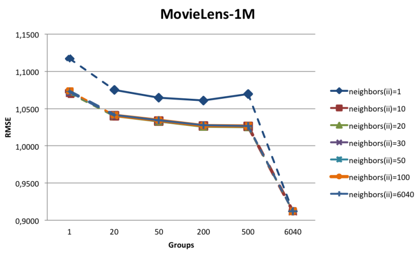

Fig. 7 RMSE values for the different values of the $neighbors^{ii}$ parameter in the MovieLens-1M dataset

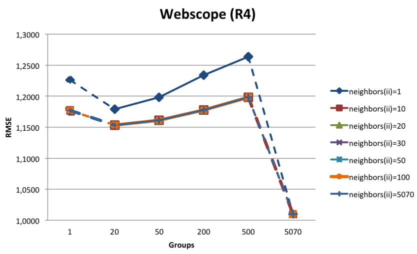

Fig. 8 RMSE values for the different values of the $neighbors^{ii}$ parameter in the Webscope (R4) dataset

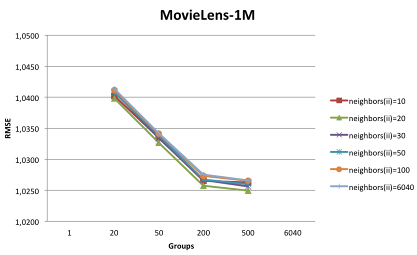

Fig. 9 Detail of the experiment to study parameter $neighbors^{ii}$ in the MovieLens-1M dataset

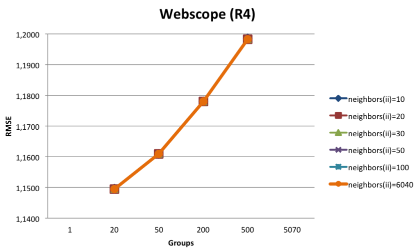

Fig. 10 Detail of the experiment to study parameter $neighbors^{ii}$ in the Webscope (R4) dataset

Fig. 7 and Fig. 8 show the accuracy for different values of $neighbors^{ii}$. In the figures it is impossible to see the one that allows a system to obtain the best results, so both were zoomed to focus on the part between 20 and 500 groups (Fig. 9 and Fig. 10). As Fig. 9 shows, in the MovieLens-1M dataset there is an improvement up to $neighbors^{ii} = 20$, then results worsen again (results are similar for $neighbors^{ii} = 10$ and $neighbors^{ii} = 30$). The RMSE values are close, so we performed the t-tests to evaluate the difference between them. The ones that compare 20 and 30 neighbors are presented in Table 13 (g is used to indicate the number of groups). There is not enough confidence to reject the null hypothesis that the results for $neighbors^{ii} = 20$ and $neighbors^{ii} = 30$ are the same. However, with $neighbors^{ii} = 20$ the probability that there is

MovieLens-1M	$RMSE_{20}$ vs. $RMSE_{30}$
g=1	$p = 0.41$
g=20	$p = 0.70$
g=50	$p = 0.63$
g=200	$p = 0.48$
g=500	$p = 0.52$

Table 13 Results of the the independent-samples two-tailed Student's t-tests that compare the accuracy obtained by employing different values of the $neighbors^{ii}$ parameter.

Webscope (R4)	$RMSE_{10}$ vs. $RMSE_{20}$
g=1	$p = 1.00$
g=20	$p = 1.00$
g=50	$p = 1.00$
g=200	$p = 1.00$
g=500	$p = 1.00$

Table 14 Results of the the independent-samples two-tailed Student's t-tests that compare the accuracy obtained by employing different values of the $neighbors^{ii}$ parameter.

a difference is between 30% and 59%, so the value used to select the items similar to the one considered is 20. Regarding the results of the Webscope (R4) dataset in Fig. 10, the difference between the results cannot be perceived. This is confirmed by the t-tests presented in Table 14, which show that there is no confidence to reject the null hypothesis that the results for $neighbors^{ii} = 10$ and $neighbors^{ii} = 20$ are the same. In order to speed up the computation of the algorithm, the value chosen for $neighbors^{ii}$ in the Webscope (R4) dataset is 10 (i.e., it is faster to predict a rating by considering less items).

5.4.3 Conclusions about the experiments on ModelBased

All the parameters used by *ModelBased* have been tested and the class will be next compared with the other classes with the following configurations:

- *Additive Utilitarian* is the strategy selected to model a group's preferences in both datasets;
- parameter *coratings* is set to 10% in both datasets;
- parameter $neighbors^{ii}$ is set to 20 in the MovieLens-1M dataset, and it is set to 10 in the Webscope (R4) dataset.

5.4.4 MergeRecommendations - Setting of the Parameters

The experiments to set $neighbors^{uu}$ and *top-n* are now presented.

Selection of the number of $neighbors^{uu}$. In order to predict a rating for a user, the set of users most similar to the one currently considered have to be chosen, by setting the $neighbors^{uu}$ parameter. Since the algorithm predicts ratings for individuals, this evaluation is done out of the group recommendation context.

Fig. 11 and Fig. 12 show the RMSE of the prediction algorithm for increasing values of $neighbors^{uu}$; this is the common way to choose the value [16]. Our results reflect the trend described by the authors, i.e., for low values of the parameter, great improvements can be noticed. As expected, RMSE takes the form of a convex function (Fig. 13 shows a detail of Fig. 11, and Fig. 14 a detail of Fig. 12), which indicates that after a certain value improvement stops. In the experiments in both datasets, that value is 100.

Independent-samples t-tests to evaluate the difference between the results obtained between 100 and the other numbers of neighbors, are presented in

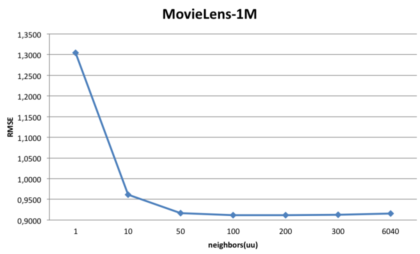

Fig. 11 RMSE values for increasing number of $neighbors^{uu}$ in the MovieLens-1M dataset

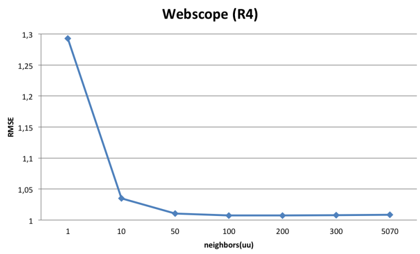

Fig. 12 RMSE values for increasing number of $neighbors^{uu}$ in the Webscope (R4) dataset

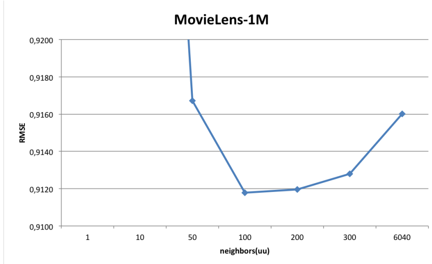

Fig. 13 RMSE takes the form of a convex function in the MovieLens-1M dataset

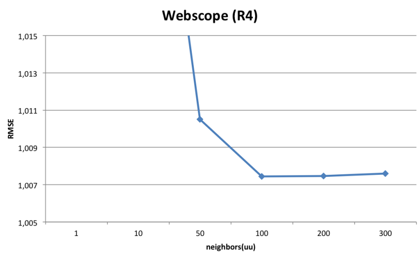

Fig. 14 RMSE takes the form of a convex function in the Webscope (R4) dataset

MovieLens-1M	p
$RMSE_1$ vs. $RMSE_{100}$	0.00
$RMSE_{10}$ vs. $RMSE_{100}$	0.00
$RMSE_{50}$ vs. $RMSE_{100}$	0.00
$RMSE_{100}$ vs. $RMSE_{200}$	0.82
$RMSE_{100}$ vs. $RMSE_{300}$	0.33
$RMSE_{100}$ vs. $RMSE_{6040}$	0.24

Table 15 Results of the independent-samples two-tailed Student's t-tests that compare the accuracy obtained by employing different values of the $neighbors^{uu}$ parameter.

Webscope (R4)	p
$RMSE_1$ vs. $RMSE_{100}$	0.00
$RMSE_{10}$ vs. $RMSE_{100}$	0.00
$RMSE_{50}$ vs. $RMSE_{100}$	0.00
$RMSE_{100}$ vs. $RMSE_{200}$	0.96
$RMSE_{100}$ vs. $RMSE_{300}$	0.83
$RMSE_{100}$ vs. $RMSE_{5070}$	0.79

Table 16 Results of the independent-samples two-tailed Student's t-tests that compare the accuracy obtained by employing different values of the $neighbors^{uu}$ parameter.

Table 15 and Table 16. Results show that in both datasets there is no difference between choosing 100 and 200 neighbors. Since it is faster to compute predictions considering 100 neighbors instead of 200, $neighbors^{uu} = 100$ is the value chosen for the algorithm.

Choice of the top-n items. *MergeRecommendations* combines the recommendations made for each user (*top-n* items). This experiment studies how big *top-n* should be, i.e., how many items to select from the predictions of a user.

Fig. 15 and Fig. 16 show that in both datasets the selection of the top-5 items leads to the best results. Independent-samples t-tests, performed to evaluate if there is a significant difference between the values obtained for the

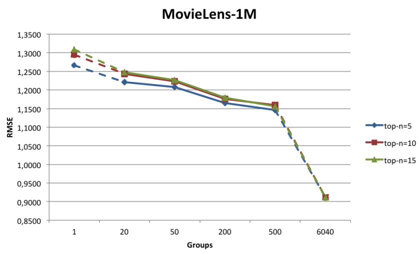

Fig. 15 RMSE values for increasing number of $neighbors^{uu}$

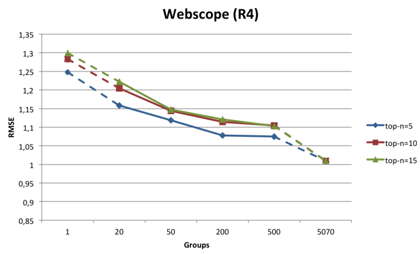

Fig. 16 RMSE takes the form of a convex function.

MovieLens-1M	$top\text{-}n = 5$ **vs.** $top\text{-}n = 10$
g=1	$p = 0.01$
g=20	$p = 0.18$
g=50	$p = 0.53$
g=200	$p = 0.60$
g=500	$p = 0.54$

Table 17 Results of the the independent-samples two-tailed Student's t-tests that compare the accuracy obtained by employing different values of *top-n* parameter.

Webscope (R4)	$top\text{-}n = 5$ **vs.** $top\text{-}n = 10$
g=1	$p = 0.00$
g=20	$p = 0.00$
g=50	$p = 0.00$
g=200	$p = 0.00$
g=500	$p = 0.00$

Table 18 Results of the the independent-samples two-tailed Student's t-tests that compare the accuracy obtained by employing different values of *top-n* parameter.

different values of *top-n*, reported that a difference exists and the tests that compare *top-n* $= 5$ and *top-n* $= 10$ (i.e., the ones that led the most similar results) are reported in Table 17 and Table 18.

Results show that when *top-n* increases, the significance of the difference between the values in the MovieLens-1M decreases. However, for *top-n* $= 5$ the highest probability that the values are not different is 40%, so the value was chosen for the system. In the Webscope (R4) we can set *top-n* $= 5$ with the confidence that the difference of the accuracy of the system with this value is significant.

5.4.5 Conclusions about the experiments on MergeRecommendation

All the *MergeRecommendations* parameters have been tested and the class will be next compared with the other classes, with the following configuration for both datasets:

- parameter $neighbors^{uu}$ is set to 100;
- parameter *top-n* is set to 5.

5.4.6 PredictionsAggregation:Cluster&Predict - Selection of the Best System

This class of systems automatically detects groups according to the constrains imposed by the system, predicts individual ratings, and models the groups.

Since the algorithm that predicts individual ratings is the same used by *MergeRecommendations*, the only remaining set of experiments is the one that

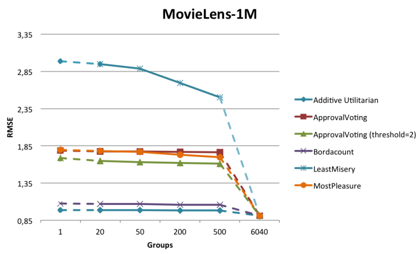

Fig. 17 RMSE values for different group modeling strategies of the *PredictionsAggregation:Cluster&Predict* class in the MovieLens-1M dataset

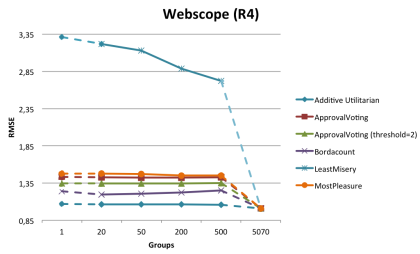

Fig. 18 RMSE values for different group modeling strategies of the *PredictionsAggregation:Cluster&Predict* class in the Webscope (R4) dataset

MovieLens-1M	$RMSE_{AU}$ **vs.** $RMSE_{BC}$
g=1	$p = 0.00$
g=20	$p = 0.00$
g=50	$p = 0.00$
g=200	$p = 0.00$
g=500	$p = 0.00$

Table 19 Results of the the independent-samples two-tailed Student's t-tests that compare the accuracy obtained by employing different group modeling strategies.

Webscope (R4)	$RMSE_{AU}$ **vs.** $RMSE_{BC}$
g=1	$p = 0.00$
g=20	$p = 0.00$
g=50	$p = 0.00$
g=200	$p = 0.00$
g=500	$p = 0.00$

Table 20 Results of the the independent-samples two-tailed Student's t-tests that compare the accuracy obtained by employing different group modeling strategies.

allows us to decide with which algorithm the groups should be modeled. Each system was run with the previously tested value of $neighbors^{uu}$, i.e., 100.

Fig. 17 and Fig. 18 show the obtained results with each system that uses a different modeling strategy. The same modeling strategies that could not be previously evaluated, could not be tested in this class too. Even for this class of systems, *Additive Utilitarian* is the strategy that best models the groups.

Independent-samples t-test have been performed to compare the results of each couple of strategies. For readability reasons, just the tests that involve the strategy closer to *Additive Utilitarian*, i.e., *Borda Count*, are presented in Table 19 and Table 20. Results clearly suggest that in both datasets *Additive Utilitarian* is the best strategy to model the groups, i.e., averaging individual ratings leads to create better models and improve the group predictions created by the system.

5.4.7 PredictionsAggregation:Predict&Cluster - Selection of the Best System

This class clusters the users by using the predictions along with the explicitly expressed ratings. So the clustering was repeated, considering also the predictions built with $neighbors^{uu} = 100$, and the modeling strategies were tested.

Fig. 19 and Fig. 20 show the RMSE values obtained by each system that uses a modeling strategy. The same modeling strategies that could not be previously evaluated, could not be tested in this class too. Results show that

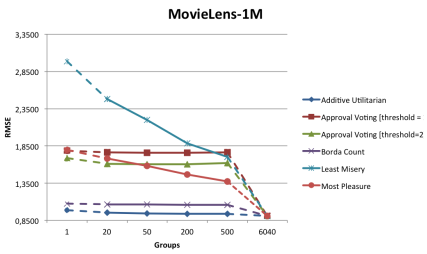

Fig. 19 RMSE values for different group modeling strategies of the *PredictionsAggregation:Predict&Cluster* class in the MovieLens-1M dataset

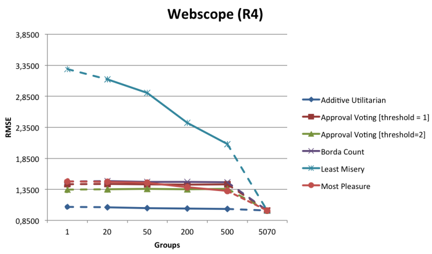

Fig. 20 RMSE values for different group modeling strategies of the *PredictionsAggregation:Predict&Cluster* class in the Webscope (R4) dataset

MovieLens-1M	$RMSE_{AU}$ **vs.** $RMSE_{BC}$
g=1	$p = 0.00$
g=20	$p = 0.00$
g=50	$p = 0.00$
g=200	$p = 0.00$
g=500	$p = 0.00$

Table 21 Results of the independent-samples two-tailed Student's t-tests that compare the accuracy obtained by different group modeling strategies.

MovieLens-1M	$RMSE_{AU}$ **vs.** $RMSE_{BC}$
g=1	$p = 0.00$
g=20	$p = 0.00$
g=50	$p = 0.00$
g=200	$p = 0.00$
g=500	$p = 0.00$

Table 22 Results of the independent-samples two-tailed Student's t-tests that compare the accuracy obtained by different group modeling strategies.

Additive Utilitarian is the strategy that allows to achieve the best results in both datasets.

Independent-samples t-test compared the results obtained by each couple of modeling strategies. The tests that involve the strategy with the accuracy closer to *Additive Utilitarian*, i.e., *Borda Count*, are presented in Table 21 and Table 22. Results clearly suggest that even for *PredictionsAggregation:Predict&Cluster*, in both datasets *Additive Utilitarian* is the best strategy to model groups, i.e., averaging individual ratings leads to produce the best group predictions with the algorithm.

5.4.8 Selection of the Best Class

Once the most performing system of each class has been chosen and parameters have been set, the accuracy values of each class can be compared.

Fig. 21 and Fig. 22 report the results of each class of systems with its best configuration. It can be observed that, for all the classes, as the number of groups grows the accuracy improves (i.e., the RMSE values get lower). As previously said, this is not true in the *ModelBased* class and the Webscope (R4) dataset (which is the smaller and sparser one), which cannot accurately predict for small groups (the 200 and 500 groups settings). In the rest of the cases, as expected, a system has a higher accuracy when more recommendation lists can be built. The three families of approaches to produce group recommendations are clearly separated in the MovieLens-1M results. Web-

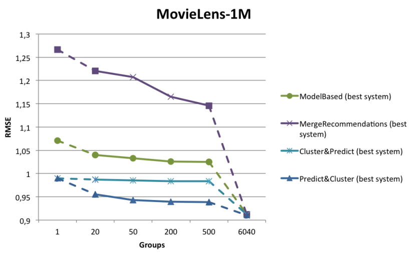

Fig. 21 RMSE values obtained by the best system of each class in the MovieLens-1M dataset. For all the classes, the best system is the one that uses the *Additive Utilitarian* strategy

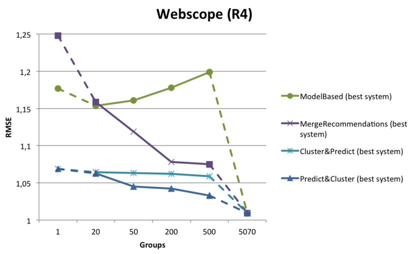

Fig. 22 RMSE values obtained by the best system of each class in the Webscope (R4) dataset. For all the classes, the best system is the one that uses the *Additive Utilitarian* strategy

scope (R4) shows a different behavior for the *ModelBased* class of systems for the reasons previously mentioned, while for the other three classes the behavior is the same of the MovieLens-1M dataset. In both cases, however the approaches that merge individual recommendations (*MergeRecommendations*) and the one based on a group model (i.e., *ModelBased*) achieve the worst results. This means that with large and automatically detected groups, if user preferences are expressed just with a small subset of items by employing an approach that merges individual recommendations (in this case five), a group recommender system is not able to properly satisfy the users. Regarding the model-based approach, these results confirm that the use of all the preferences expressed by a user are needed, in order to build accurate group recommendations. At the bottom of the figures, achieving the best results, are the classes that merge individual preferences (i.e., *PredictionsAggregation:Cluster&Predict* and *PredictionsAggregation:Predict&Cluster*). Moreover, the accuracy of *PredictionsAggregation:Predict&Cluster* is much higher than the one of *PredictionsAggregation:Cluster&Predict*, and this proves that enhancing the clustering with individual predictions leads to improvements in the predicted results' accuracy.

Independent-samples t-tests, performed to compare the results, confirm that there is a significant difference between the RMSE values obtained by *PredictionsAggregation:Predict&Cluster* and the ones obtained with the other systems. These results are not presented to facilitate the reading of the paper.

5.5 Discussion

In this section, a summary of the results of our experiments is given, in the form of a set of best practices aimed at a researcher or a software designer involved in real world scenarios where the number of recommendation lists that can be produced is limited. More precisely, the scenario is supposed to have a large number of users and a small number of recommendation lists with respect to the number of users. An example could be the fliers application scenario presented in the Introduction.

The main questions that arise when designing a system are the following:

- how should the preferences for the items not rated be predicted?
- how should the users be grouped, in order to obtain groups of users with similar preferences?
- how should the preferences of a group be modeled, in order to derive a rating for each item for the group?

Some answers coming from the results of our experiments are the following:

- When groups are automatically detected, it is better to predict the preferences for the missing items for each user. Allowing the system to aggregate a preference for each user and each item improves the accuracy of a group recommender system, with respect to other approaches that consider only the preferences expressed by the users. In order to have accurate predictions for each user, we found that a User-Based Collaborative Filtering approach (presented in 4.2) is the best option. A good sizing of the neighbors considered by the algorithm is 100. In fact, as the experiments performed on the algorithm show (see 5.4.4), results take the form of a convex function. The accuracy improves when the number of neighbors increases up to about 100. With higher numbers of neighbors, the accuracy worsens.
- Since in an application scenario like the one considered in this paper groups do not exist, in order to find them a form of unsupervised classification (*clustering*) is necessary. We found that it is best to cluster the users based on the preferences for the items (see 5.4.8). In order to produce groups that actually have homogeneous preferences, as much information as possible should be given to the clustering algorithm, so the predictions are added to the input of the algorithm. The results presented in 5.4.8 show that adding individual predictions to the clustering algorithm input leads to great improvements in the performance of a group recommender system.
- A group model created by calculating an average of the individual preferences (i.e., by using the *Additive Utilitarian* modeling strategy) has to be preferred with respect to the other strategies (see 5.4.1, 5.4.6, 5.4.7). Since the system deals with a limited number of recommendation lists, groups tend to be large and an average is a value that can be considered representative of the whole group. Moreover, for groups created with the k-means clustering algorithm, creating a group model with an average of the individual values for each item is like re-creating the centroid of the cluster, i.e., a super-user that connects every user of the group (in fact the k-means clustering algorithm minimizes the *average distortion*, i.e., the mean squared distance from each data point to its nearest center).

More considerations can be done when particular conditions are present:

- If groups are not homogeneous in terms of size, different strategies could be used to model each group. For example, if a group is very small, a strategy

like *Least Misery* (usually adopted by group recommendation approaches that work with small groups) could be tested in this scenario.
- If there is a user that can be considered as a "centroid" for the group (i.e., her/his distance to the centroid of the cluster is very small), there is no need to model the group calculating an average of the individual ratings (*Additive Utilitarian* strategy), since her/his ratings can be considered as representative of the group. So, a strategy like *Most Respected Person*, which considers the ratings of a person that leads the group, could be used in order to reduce the computational time of the modeling step.
- If additional information about the users is available and that information is related to the application scenario in which the system operates, that information could be used to form the groups (for example, a notion of friendship could be exploited by a system that recommends travels).

A few more considerations can be done on what should be avoided:

- Group recommendations should not be produced by merging individual recommendations (see 5.4.8). In fact, each user would be represented by a limited amount of preferences (the results in 5.4.4 illustrated that the top-5 predicted items should be recommended to the group). Moreover, only high ratings would contribute to the group model, so a group recommender system would not have any information about what each user does not like.
- In a context in which groups are automatically detected, predictions for the items not rated should not be built for groups (see 5.4.8). In fact, group predictions are built by using a model that aggregates the individual preferences. If groups are large, the aggregate ratings are not representative of the users in the group and the accuracy of the system is affected.
- When modeling a large and automatically detected group, strategies that advantage a small part of the group to model the group preferences should be avoided (see 5.4.1, 5.4.6, 5.4.7). This is also true for strategies that use a *threshold* value (i.e., strategies that do not consider the ratings under/above a certain value). In fact, using these strategies, a user is not considered or a part of her/his preferences is not included in the model, and the effectiveness of the group recommendation for the users is affected.

6 Conclusions and Future Work

This paper presented *ART (Automatic Recommendation Technologies)*, a set of group recommender systems that automatically detect groups of users with similar preferences, in order to respect the constraints imposed by the system on the number of recommendation lists that can be produced.

The extensive analysis performed on the proposed systems allowed us to achieve four strong scientific contributions, now recapped.

- When recommendations have to be produced for groups that are automatically detected, group preferences for an item should be modeled with an

average of the individual values (*Additive Utilitarian* strategy). In fact, when the number of recommendation lists that can be produced is limited, groups tend to be large and a strategy that advantages a small portion of the group or a user tend to worsen the performance of an algorithm.
– The approaches that use all the predictions built for the individual users are more accurate, with respect to the ones that build specific predictions for a group or use only the items with the highest predicted rating, which are not properly able to model the preferences of each user in a group.
– The use of individual predictions as input for the clustering algorithm allows a system to achieve better results with respect to the approach that clusters users using only the explicitly expressed preferences. This means that the additional information provided to the clustering algorithm is useful in order to produce better group recommendations, since the matrix is not sparse and the additional data given to the clustering algorithm is produced by an algorithm that has been largely evaluated and is able to produce predictions that closely reflect the preferences of a user.
– A set of best practices aimed at helping in the development of a real world group recommender system that automatically detects groups was given.

Recent studies [26–28] showed that behavioral data mining is an effective tool to connect the users of a social network. Future work will study our group recommendation approach in contexts in which the social behavior of the users can be exploited, in order to cluster the users based on the behavioral features.

References

1. Amatriain, X., Jaimes, A., Oliver, N., Pujol, J.M.: Data mining methods for recommender systems. In: Recommender Systems Handbook, pp. 39–71. Springer, Boston, MA (2011)
2. Ardissono, L., Goy, A., Petrone, G., Segnan, M., Torasso, P.: Intrigue: Personalized recommendation of tourist attractions for desktop and hand held devices. Applied Artificial Intelligence **17**(8-9), 687–714 (2003)
3. Baatarjav, E.A., Phithakkitnukoon, S., Dantu, R.: Group recommendation system for facebook. In: R. Meersman, Z. Tari, P. Herrero (eds.) On the Move to Meaningful Internet Systems: OTM 2008 Workshops, *Lecture Notes in Computer Science*, vol. 5333, pp. 211–219. Springer Berlin Heidelberg (2008)
4. Baltrunas, L., Makcinskas, T., Ricci, F.: Group recommendations with rank aggregation and collaborative filtering. In: Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, pp. 119–126. ACM, New York, NY, USA (2010)
5. Bellman, R.E.: Adaptive control processes - A guided tour. Princeton University Press, Princeton, New Jersey, U.S.A. (1961)
6. Boratto, L., Carta, S.: State-of-the-art in group recommendation and new approaches for automatic identification of groups. In: Information Retrieval and Mining in Distributed Environments, *Studies in Computational Intelligence*, vol. 324, pp. 1–20. Springer Berlin Heidelberg (2011)
7. Boratto, L., Carta, S., Chessa, A., Agelli, M., Clemente, M.L.: Group recommendation with automatic identification of users communities. In: Proceedings of the 2009 IEEE/WIC/ACM International Conference on Web Intelligence and International Conference on Intelligent Agent Technology - Workshops, pp. 547–550. IEEE (2009)
8. Boratto, L., Carta, S., Satta, M.: Groups identification and individual recommendations in group recommendation algorithms. In: Practical Use of Recommender Systems, Algorithms and Technologies 2010, *CEUR Workshop Proceedings*, vol. 676 (2010)

9. Bourke, S., McCarthy, K., Smyth, B.: Using social ties in group recommendation. In: AICS 2011: Proceedings of the 22nd Irish Conference on Artificial Intelligence and Cognitive Science: 31 August-2 September, 2011: University of Ulster-Magee. Intelligent Systems Research Centre (2011)
10. Carolis, B.D., Pizzutilo, S.: Providing relevant background information in smart environments. In: E-Commerce and Web Technologies, 10th International Conference, EC-Web 2009. Proceedings, *Lecture Notes in Computer Science*, vol. 5692, pp. 360–371. Springer (2009)
11. Carvalho, L.A.M., Macedo, H.T.: Generation of coalition structures to provide proper groups' formation in group recommender systems. In: Proceedings of the 22Nd International Conference on World Wide Web Companion, WWW '13 Companion, pp. 945–950. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2013)
12. Chen, Y., Pu, P.: Cofeel: Emotional social interface in group recommender systems. In: Proceedings of the RecSys'12 Workshop on Interfaces for Recommender Systems, in conjunction with the 6th ACM Conference on Recommender Systems (RecSys'12), pp. 48–55 (2012)
13. Chen, Y., Pu, P.: Cofeel: Using emotions to enhance social interaction in group recommender systems. In: Alpine Rendez-Vous (ARV) 2013 Workshop on Tools and Technology for Emotion-Awareness in Computer Mediated Collaboration and Learning (2013)
14. Christensen, I.A., Schiaffino, S.N.: Entertainment recommender systems for group of users. Expert Systems with Applications **38**(11), 14,127–14,135 (2011)
15. Crossen, A., Budzik, J., Hammond, K.J.: Flytrap: intelligent group music recommendation. In: IUI, pp. 184–185 (2002)
16. Desrosiers, C., Karypis, G.: A comprehensive survey of neighborhood-based recommendation methods. In: Recommender Systems Handbook, pp. 107–144. Springer, Berlin (2011)
17. Felfernig, A., Zehentner, C., Ninaus, G., Grabner, H., Maalej, W., Pagano, D., Weninger, L., Reinfrank, F.: Group decision support for requirements negotiation. In: Advances in User Modeling - UMAP 2011 Workshops, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 7138, pp. 105–116. Springer (2012)
18. Herlocker, J., Konstan, J., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: Research and Development in Information Retrieval. American Association of Computing Machinery, American Association of Computing Machinery (1999)
19. Jameson, A.: More than the sum of its members: challenges for group recommender systems. In: Proceedings of the working conference on Advanced visual interfaces, AVI 2004, pp. 48–54. ACM Press (2004)
20. Jameson, A., Baldes, S., Kleinbauer, T.: Two methods for enhancing mutual awareness in a group recommender system. In: Proceedings of the working conference on Advanced visual interfaces, AVI 2004, pp. 447–449. ACM Press (2004)
21. Jameson, A., Smyth, B.: Recommendation to groups. In: The Adaptive Web, Methods and Strategies of Web Personalization, *Lecture Notes in Computer Science*, vol. 4321, pp. 596–627. Springer, Berlin (2007)
22. Jung, J.J.: Attribute selection-based recommendation framework for short-head user group: An empirical study by movielens and imdb. Expert Systems with Applications **39**(4), 4049–4054 (2012). DOI 10.1016/j.eswa.2011.09.096
23. Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R., Wu, A.Y.: An efficient k-means clustering algorithm: Analysis and implementation. IEEE Trans. Pattern Anal. Mach. Intell. **24**, 881–892 (2002). DOI 10.1109/TPAMI.2002.1017616
24. Lieberman, H., Dyke, N.W.V., Vivacqua, A.S.: Let's browse: A collaborative web browsing agent. In: IUI, pp. 65–68 (1999)
25. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press (1967)
26. Manca, M., Boratto, L., Carta, S.: Design and architecture of a friend recommender system in the social bookmarking domain. In: Proceedings of the Science and Information Conference 2014, pp. 838–842 (2014)

27. Manca, M., Boratto, L., Carta, S.: Mining user behavior in a social bookmarking system-a delicious friend recommender system. In: Proceedings of the 3rd International Conference on Data Management Technologies and Applications (DATA 2014), pp. 331–338 (2014)
28. Manca, M., Boratto, L., Carta, S.: Friend recommendation in a social bookmarking system: Design and architecture guidelines. In: K. Arai, S. Kapoor, R. Bhatia (eds.) Intelligent Systems in Science and Information 2014, *Studies in Computational Intelligence*, vol. 591, pp. 227–242. Springer International Publishing (2015)
29. Masthoff, J.: Modeling a group of television viewers. In: Proceedings of the Future TV: Adaptive Instruction In Your Living Room (A workshop for ITS 2002) (2002)
30. Masthoff, J.: Group modeling: Selecting a sequence of television items to suit a group of viewers. User Modeling and User-Adapted Interaction **14**(1), 37–85 (2004). DOI 10.1023/B:USER.0000010138.79319.fd
31. Masthoff, J.: The pursuit of satisfaction: affective state in group recommender systems. In: Proceedings of the 10th international conference on User Modeling, UM'05, pp. 297–306. Springer-Verlag, Berlin, Heidelberg (2005)
32. Masthoff, J.: Group Adaptation and Group Modelling. In: Intelligent Interactive Systems in Knowledge-Based Environments, *Studies in Computational Intelligence*, vol. 104, pp. 157–173. Springer (2008)
33. Masthoff, J.: Group recommender systems: Combining individual models. In: Recommender Systems Handbook, pp. 677–702. Springer (2011)
34. Masthoff, J., Gatt, A.: In pursuit of satisfaction and the prevention of embarrassment: affective state in group recommender systems. User Modeling and User-Adapted Interaction **16**(3-4), 281–319 (2006). DOI 10.1007/s11257-006-9008-3
35. McCarthy, J.F.: Pocket restaurantfinder: A situated recommender system for groups. In: Workshop on Mobile Ad-Hoc Communication at the 2002 ACM Conference on Human Factors in Computer Systems. Minneapolis (2002)
36. McCarthy, J.F., Anagnost, T.D.: Musicfx: An arbiter of group preferences for computer supported collaborative workouts. In: CSCW '98, Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work, pp. 363–372. ACM (1998)
37. McCarthy, K., Salamó, M., Coyle, L., McGinty, L., Smyth, B., Nixon, P.: Cats: A synchronous approach to collaborative group recommendation. In: Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference, pp. 86–91. AAAI Press (2006)
38. McCarthy, K., Salamó, M., Coyle, L., McGinty, L., Smyth, B., Nixon, P.: Group recommender systems: a critiquing based approach. In: Proceedings of the 2006 International Conference on Intelligent User Interfaces, pp. 267–269. ACM (2006)
39. Ntoutsi, E., Stefanidis, K., Nørvåg, K., Kriegel, H.P.: Fast group recommendations by applying user clustering. In: Conceptual Modeling - 31st International Conference ER 2012. Proceedings, *Lecture Notes in Computer Science*, vol. 7532, pp. 126–140. Springer (2012)
40. Ntoutsi, I., Stefanidis, K., Nørvåg, K., Kriegel, H.P.: grecs: A group recommendation system based on user clustering. In: Database Systems for Advanced Applications - 17th International Conference, DASFAA 2012, Proceedings, Part II, *Lecture Notes in Computer Science*, vol. 7239, pp. 299–303. Springer (2012)
41. O'Connor, M., Cosley, D., Konstan, J.A., Riedl, J.: Polylens: A recommender system for groups of users. In: Proceedings of the Seventh European Conference on Computer Supported Cooperative Work, pp. 199–218. Kluwer (2001)
42. Padmanabhan, V., Seemala, S.K., Bhukya, W.N.: A rule based approach to group recommender systems. In: Proceedings of the 5th International Conference on Multi-Disciplinary Trends in Artificial Intelligence, MIWAI'11, pp. 26–37. Springer-Verlag, Berlin, Heidelberg (2011)
43. Pessemier, T., Dooms, S., Martens, L.: Comparison of group recommendation algorithms. Multimedia Tools and Applications pp. 1–45 (2013). DOI 10.1007/s11042-013-1563-0
44. Ricci, F.: Recommender systems: Models and techniques. In: Encyclopedia of Social Network Analysis and Mining, pp. 1511–1522. Springer New York (2014)
45. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Recommender Systems Handbook, pp. 1–35. Springer, Berlin (2011)

46. Sánchez, L.Q., Bridge, D.G., Díaz-Agudo, B., Recio-García, J.A.: A case-based solution to the cold-start problem in group recommenders. In: Case-Based Reasoning Research and Development - 20th International Conference, ICCBR 2012. Proceedings, *Lecture Notes in Computer Science*, vol. 7466, pp. 342–356. Springer (2012)
47. Schafer, J.B., Frankowski, D., Herlocker, J.L., Sen, S.: Collaborative filtering recommender systems. In: The Adaptive Web, Methods and Strategies of Web Personalization, *Lecture Notes in Computer Science*, vol. 4321, pp. 291–324. Springer (2007)
48. Senot, C., Kostadinov, D., Bouzid, M., Picault, J., Aghasaryan, A.: Evaluation of group profiling strategies. In: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, pp. 2728–2733. IJCAI/AAAI (2011)
49. Senot, C., Kostadinov, D., Bouzid, M., Picault, J., Aghasaryan, A., Bernier, C.: Analysis of strategies for building group profiles. In: User Modeling, Adaptation, and Personalization, 18th International Conference, UMAP 2010. Proceedings, *Lecture Notes in Computer Science*, vol. 6075, pp. 40–51. Springer (2010)
50. Zha, Z.J., Tian, Q., Cai, J., Wang, Z.: Interactive social group recommendation for flickr photos. Neurocomput. **105**, 30–37 (2013). DOI 10.1016/j.neucom.2012.06.039. URL http://dx.doi.org/10.1016/j.neucom.2012.06.039