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

The Rating Prediction Task in a Group Recommender System that Automatically Detects Groups: Architectures, Algorithms, and Performance Evaluation

Ludovico Boratto · Salvatore Carta

Received: date / Accepted: date

Abstract A recommender system suggests items to users by *predicting* what might be interesting for them. The prediction task has been highlighted in the literature as the most important one computed by a recommender system. Its role becomes even more central when a system works with groups, since the predictions might be built for each user or for the whole group. This paper presents a deep evaluation of three approaches, used for the prediction of the ratings in a group recommendation scenario in which groups are detected by clustering the users. Experimental results confirm that the approach to predict the ratings strongly influences the performance of a system and show that building predictions for each user, with respect to building predictions for a group, leads to great improvements in the accuracy of the recommendations.

Keywords Group Recommendation · Clustering · Rating Prediction

1 Introduction

A recommender system suggests items that might be interesting for a user. In order to identify the useful items for the user, a recommender system must *predict* that an item is worth recommending [25]. The prediction is based on an estimation of the utility of the items for a user and allows the system to recommend the items with the highest utility. As highlighted in [25, 1], the prediction task is the recommendation computation core.

Group recommendation is designed for contexts in which more than a person is involved in the recommendation process [3, 14]. Group recommender

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

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

systems suggest items to a group, by combining individual models that contain a user's preferences [20].

Rating prediction in a group recommender system. According to Jameson and Smyth [14], a system can generate group recommendations by using three different approaches to predict the ratings:

- construction of group preference models and prediction of the missing ratings for each group by using the model;
- prediction of the ratings for the items not rated by each user and merging of the individual recommendations made for the members of a group;
- aggregation of the predictions built for every user into a group preference;

From the previous points it can be noticed that the rating prediction task takes a central role even in a group recommender system. In fact, the ratings for the missing items might be predicted for each user or for the group. So, according to the approach chosen to predict the ratings, the architecture of the system changes. In fact, based on the approach used, the prediction task takes a different input (i.e., a group model or the individual preferences expressed by each user) and produces a completely different output (i.e., predictions for a group, predictions for each user, or recommendations for each user). This means that the flow of the computation radically changes, in order to allow the system to build the predictions. Moreover, the results produced by the system (i.e., the group recommendations) are significantly different.

Group recommendation and detection of groups. A particular application scenario in which group recommendation is useful is when the number of recommendation lists that can be generated by the system is limited (i.e., it is not possible to recommend a list of items to each user).

> *A company decides to print recommendation flyers that present suggested products. Even if the data to produce a flyer with individual recommendations for each customer is available, 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.*

With respect to classic group recommendation, this type of systems add the complexity of optimally defining groups, in order to automatically respect the constraint on the number of recommendation lists that can be produced and maximize users' satisfaction.

The presence of a classification task inside the group recommendation process, which allows to detect the groups, is a novel aspect that is not present in the existing systems in the literature. Studying this research area becomes even more challenging with respect to classic group recommendation, because we deal both with the classification and recommendation aspects, in order to be able produce the recommendations.

Our contributions. This paper explores the rating prediction task in the previously mentioned scenario, i.e., a group recommender system that, given

a limited number of recommendation lists that can be produced, is able to detect a number of groups equal to the number of recommendation lists.

The objective of this work is to identify the best approach to predict the ratings for a group that has been automatically detected by the system. Three different recommender systems have been developed in order to produce the predictions according to the previously mentioned three approaches. The systems have been tested and their performance have been compared in order to identify the approach that allows to predict the most accurate ratings.

This study is motivated by the recognized importance of the rating prediction task in the literature, by the fact that groups are automatically detected by the system, and by the need to produce accurate recommendations able to satisfy large groups.

The scientific contributions of our proposal are the following:

- since no work in the literature is able to automatically adapt to constraints imposed by the system on the number of recommendation lists, this is the first time that the prediction task is explored in a scenario in which groups are automatically detected;
- this paper is the first work where the three approaches to build the predictions in a group recommender systems are compared;
- for each developed system, the trade-off between the number of groups that are detected and the accuracy of the system is explored, allowing the quantitative evaluation of how the number of groups affects the performance of the system.

This paper extends the work presented in [4] in the following ways: (i) the overall description of our approach has been enhanced; (ii) related work has been extended by illustrating how the existing works in the group recommendation literature predict the ratings; (iii) a novel statistical evaluation of the results, in order to deepen the comparison of the proposed systems, has been added; (iv) a discussion of the results is introduced.

The rest of the paper is structured as follows. Section 2 describes, for each approach to predict the ratings in a group recommender system, the algorithm used to predict the ratings, the architecture of a system that uses the considered approach, and the main works in the literature that use it. Section 3 presents the three group recommender systems that automatically detect groups, developed to use the three approaches to build the predictions. Section 4 illustrates the experiments performed on the systems and presents the obtained results. Section 5 contains comments, conclusions, and future work.

2 Rating Prediction in Group Recommendation: Background and Related Work

Given a set of individual preferences, expressed in the form of ratings for the items that a user evaluated, group preferences can be generated by using one

of three approaches [14]: (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 all the predictions built for every user.

Each family of approaches will now be described in detail, along with the architecture of the system and references to the main approaches in the literature that use it.

2.1 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 not rated by the group using that model. It follows two main steps, described below.

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

The architecture of a system that uses this approach is depicted in Fig. 1 (note that the prediction task is highlighted in the figure). As previously described, in order to build the predictions for a group, the system has to produce a group model that contains the preferences of the group (TASK 1). The task that builds the model receives as input the ratings for the item evaluated by each user (INPUT 1) and the composition of each group (INPUT 2). Each group model is used to predict the ratings for the group (TASK 2). After the system has predicted the ratings, group recommendations are selected.

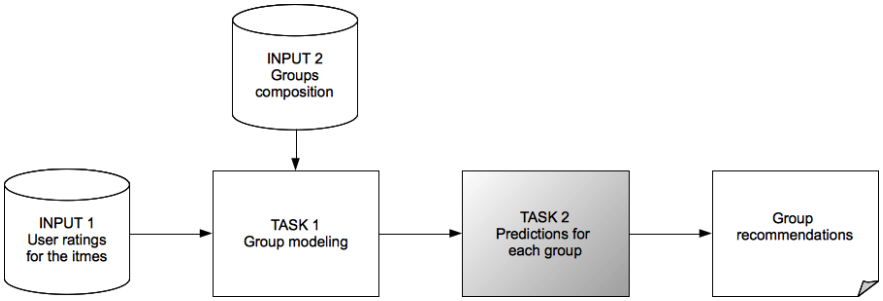

Fig. 1 Architecture of a system that uses group preference models to build the predictions

In the following, we will describe the main approaches in the literature that use this approach.

MusicFX [22] is a system that recommends music to the members of a fitness center. Since people in the room change continuously, the system gives the users that are working out in the fitness center the possibility to login.

The music to play is selected by considering the preferences of each user in a summation formula.

Flytrap [9] similarly selects music to be played in a public room. A 'virtual DJ' agent is used by the system to automatically decide the song to play. The agent analyzes the MP3 files played by a user in her/his computer and considers the information available about the music (like similar genres, artists, etc.). The song to play is selected through a voting system, in which an agent represents each user in the room and rates the candidate tracks.

In-Vehicle Multimedia Recommender [29] is a system that aims at selecting multimedia items for a group of people traveling together. The system aggregates the profiles of the passengers and merges them by using a notion of distance between the profiles. A content-based system is used to compare multimedia items and group preferences.

FIT (Family Interactive TV System) [11] is a TV program recommender system. The only input required by the system is a stereotype user representation (i.e., a class of viewers that would suit the user, like *women*, *businessmen*, *students*, etc.), along with the user preferred watching time. When someone starts watching TV, the system looks at the probability of each family member to watch TV in that time slot and predicts who there might be watching TV. Programs are recommended through an algorithm that combines such probabilities and the user preferences.

TV4M [28] recommends TV programs for multiple viewers. The system identifies who is watching TV by providing a login feature. In order to build a group profile that satisfies most of its members, all the current viewers profiles are merged, by doing a total distance minimization of the features available (e.g., genre, actor, etc.). According to the built profile, programs are recommended to the group.

In [23] a group recommender system called *CATS (Collaborative Advisory Travel System)* is presented. Its aim is to help a group of friends plan and arrange ski holidays. To achieve the objective, users are positioned around a device called "DiamondTouch table-top" and the interactions between them (since they physically share the device) help the development of the recommendations.

2.2 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 member of the group. The approach works as follows.

1. For each member of the group u:
   - For each item i not rated by the user, predict a rating p_{ui}.
   - Select the set C_i of items with the highest predicted ratings p_{ui} for u.
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.

The architecture of a system that uses this approach is illustrated in Fig. 2. The system uses the ratings for the item evaluated by each user (INPUT 1) to predict the ratings for the missing items for each user (TASK 1). The output produced by the task (the top-n predictions, which represent the recommendations for a user), are given as input to the task that merges the recommendations (TASK 2) along with the composition of each group (INPUT 2). After the system has merged the individual recommendations, group recommendations are selected.

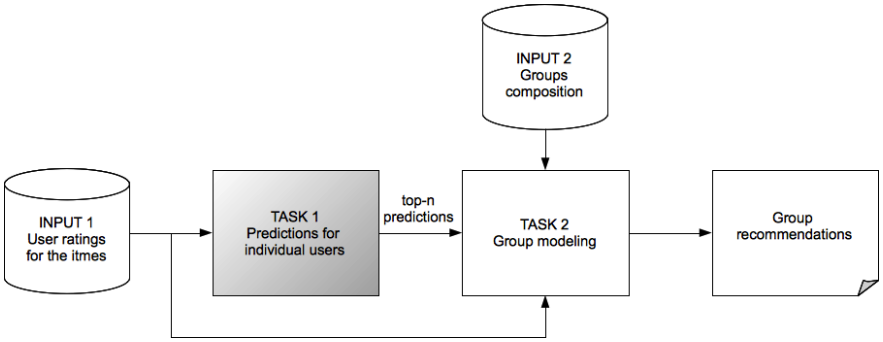

Fig. 2 Architecture of a system that merges the recommendations produced for each user

This approach is not widely used in literature. The main relevant work that embraces it is *PolyLens* [24], a system built to produce recommendations for groups of users who want to see a movie. A Collaborative Filtering approach is used to produce recommendations for each user of the group. The movies with the highest recommended ratings are considered and a "least misery" strategy is used, i.e., the recommended rating for a group is the lowest predicted rating for a movie, to ensure that every member is satisfied.

2.3 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 to derive a group preference. The approach works as follows:

1. For each item i:
   - For each member 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 members of the group, either expressed (r_{ui}) or predicted (p_{ui}).

The architecture of a system that uses this approach is shown in Fig. 3. The system works similarly to the previously presented approach. The ratings

for the items evaluated by each user (INPUT 1) are used as input by the task that predicts the ratings for the missing items for each user (TASK 1). The output produced by the task (all the calculated predictions) is given as input along with the ratings given by the users for the items (INPUT 1) and the composition of each group (INPUT 2) to the task that models the group preferences (TASK 2). After the system has modeled the preferences of each group, group recommendations are selected.

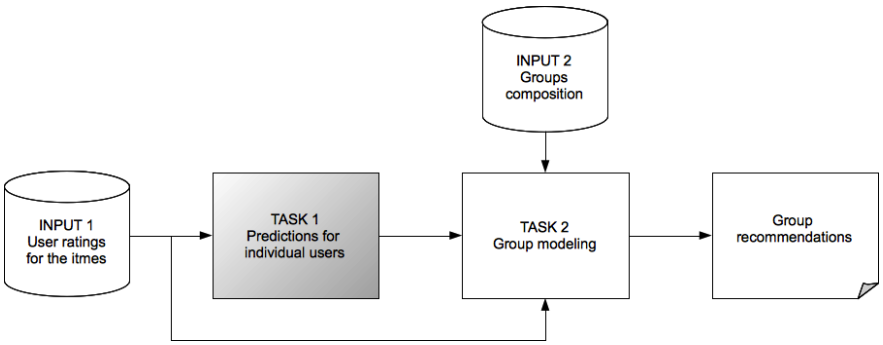

Fig. 3 Architecture of a system that aggregates individual predictions

In the following, we will describe the main approaches in the literature that use this approach.

Pocket RestaurantFinder [21] is a system that suggests restaurants to groups of people who want to dine together. Each user fills a profile with preferences about restaurants, like the price range or the type of cuisine they like (or do not like). Once the group composition is known, the system estimates individual preference for each restaurant and averages those values to build a group preference and produce a list of recommendations.

Travel Decision Forum [13] is a system that helps groups of people plan a vacation. Since the system aims at finding an agreement between the members of a group, asynchronous communication is possible and, through a web interface, a member can view (and also copy) other members' preferences. Recommendations are made by using the median of the individual preferences.

In [8], Chen and Pu present *CoFeel*, an interface that allows to express through colors the emotions given by a song chosen by the *GroupFun* music group recommender system. The interface allows users to give a feedback about how much they liked the song and the system considers the preferences expressed through the emotions, in order to generate a playlist for a group.

In [15], Jung develops an approach to identify long tail users, i.e., users who can be considered as experts on a certain attribute. So, the ratings given by the long tail user groups are used, in order to provide a relevant recommendation to the non-expert user groups, which are called short head groups.

Choicla [27] is a group recommender systems that aims at combining individual decisions into a group decision, independently from the domain in which the system operates. In order to do so, different state-of-the-art aggregation strategies are adopted.

3 Predicting Ratings for Automatically Detected Groups

As highlighted in the Introduction, no group recommendation approach is able to detect groups in order to overcome limits imposed by the system on the number of recommendation lists that can be produced.

This section presents three group recommender systems able to produce suggestions and respect a constraint on the number of recommendation lists that can be generated. Each system implements one of the three approaches to predict the ratings described in the previous section. The other tasks are implemented in the same way in all the systems, in order to evaluate how each approach to predict the ratings affects the performance of a group recommender system. All the proposed algorithms are based on state-of-the-art solutions, which have been specifically adapted for this application scenario. This choice was made in order to introduce novelty to our approach (none of the group recommender systems performs the tasks in the same way the state-of-the-art does), while exploiting the effectiveness of the existing solutions.

3.1 System Based on Group Model Construction

ModelBased is a group recommender system that detects groups of similar users, models each group by using the preferences of its members, and predicts group preferences using the model, according to the approach presented in 2.1.

Here, we describe the tasks performed by the systems and how they have been implemented.

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 [17].
2. *Group modeling.* Once groups have been detected, a group model is built for each group g, by using the *Additive Utilitarian* modeling strategy. For each group, a rating is calculated for a subset of items.
3. *Predictions for Each Group.* A group rating is predicted for each item not modeled by the previous task, by using the model that contains its preferences, with an Item-based Collaborative Filtering approach presented in [26].

All the tasks are now going to be described in detail.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u	r_{u1}		r_{u3}				r_{u7}	r_{u8}

Table 1 Example of *user model*

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	r_{11}		r_{13}	r_{14}			r_{17}	r_{18}
u_2	r_{21}	r_{22}		r_{24}	r_{25}		r_{27}	
u_3		r_{22}	r_{33}	r_{34}		r_{36}		
				...				
u_n	r_{n1}	r_{n2}	r_{n3}		r_{n5}	r_{n6}		r_{n8}

Table 2 Example of *rating matrix*

3.1.1 Detection of the Groups (Clustering)

In order to respect the constraint on the number of recommendation lists that can be produced, the set of users has first to be partitioned into a number of groups equal to the number of recommendations. Moreover, the preferences of the users in a group have to be homogeneous, in order to produce group recommendations that satisfy the individual users that belong to the group. As mentioned in the Introduction, in an application scenario like the one considered in this paper groups do not exist and, in order to find them, a form of unsupervised classification (*clustering*) is necessary.

In order to create groups with similar preferences, a *user model*, like the one in Table 1, is considered. A model contains, for each item i_n that a user u evaluated, a rating r_{un}, which expresses with a numerical value how much the user likes the item. All the user models like the one previously described take the form of a matrix, usually called *rating matrix*, like the one in Table 2.

In [2], the authors highlight that the k-means clustering algorithm is by far the most used clustering algorithm in recommender systems and alternatives are rarely used, mostly because of the simplicity and the efficiency that the algorithm can offer. Moreover, in previous studies we analyzed [6] and compared [7] 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 in this context.

This task detects groups by clustering users with the k-means clustering algorithm, which takes as input a rating matrix, like the one presented in Table 2. So, users are grouped based on the ratings available in the individual user models (each user model is represented as a line in the rating matrix and contains her/his preferences).

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.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	8	10	7	10	9	8	10	6
u_2	7	10	6	9	8	10	9	4
u_3	5	1	8	6	9	10	3	5
Group	20	21	21	25	26	28	22	15

Table 3 Example of the *Additive Utilitarian* strategy

3.1.2 Group modeling

In order to create a model that represents the preferences of a group, the *Additive Utilitarian* group modeling strategy [20] is adopted. The strategy sums the individual ratings for each item and produces a list of group ratings (the higher the sum is, the earlier the item appears in the list). The ranked list of items is exactly the same that would be produced when averaging the individual ratings, so this strategy is also called 'Average strategy'. An example of how it works is given in Table 3. The example considers three users (u_1, u_2 and u_3) that rate eight items (i_1, ..., i_8) with a rating from 1 to 10.

In a recent work [5] we showed that this metric allows to obtain the most accurate performance for a group recommender system that detects groups. Indeed, results show that:

- since the considered scenario deals with a limited number of recommendation lists, the system works with large groups. Therefore, an average, which is a single value that is meant to typify a set of different values, is the best way to put together the ratings in this context;
- 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 recreating the centroid of the cluster, i.e., a super-user that connects every user of the group.

In order to have the same scale of ratings both in the group models and in the individual user models, the produced group models contain the average of the individual predictions, instead of the sum.

In order to calculate meaningful ratings for a group g, an aggregate score r_{gi} is a part of 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 score cannot be considered representative of the preferences of the group as a whole and should not be a part of the model. So, a parameter named *coratings*, is set. The parameter expresses the minimum percentage of group members who have to rate an item, in order to include the aggregate score in the model.

3.1.3 Predictions for Each Group

In the group models created by the previous task, for a subset of items there is no preference. In order to estimate such preferences, a rating p_{gi} for an item i not available in the model of a group g is predicted through the model that

contains a group's preferences. This is done by using an Item-Based Nearest Neighbor Collaborative Filtering algorithm presented in [26]. The choice of using an Item-Based approach is made because the algorithm deals with group models. Since groups might be very large, a group model might put together a lot of preferences and it would not be significant to make a prediction with a User-based approach that would look for "similar groups" (this type of similarity would not be accurate enough to make predictions).

The algorithm predicts a rating p_{gi} for each item i not evaluated by enough users of a group g, by considering the rating r_{gj} of each similar item j rated by the group. Equation 1 gives the formula used to predict the ratings:

$$p_{gi} = \frac{\sum_{j \in ratedItems(g)} itemSim(i,j) \cdot r_{gj}}{\sum_{j \in ratedItems(g)} itemSim(i,j)} \tag{1}$$

According to [26], some authors do not consider all the items in the model (i.e., the set $ratedItems(g)$), but just the top n correlations. In order to reduce the computational complexity of the algorithm and select only the most meaningful correlations, this is the approach used for this task.

In order to compute the similarity $itemSim(i,j)$ between two items, adjusted-cosine similarity is used. The metric is the most popular measure and believed to be the most accurate when calculating similarities between items [26]. It is computed by considering all users who rated both item i and item j. Equation 2 gives the formula for the similarity (U_{ij} is the set of users that rated both item i and j).

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

3.2 System that Merges Individual Recommendations

MergeRecommendations is a group recommender system that detects groups of similar users, predicts individual preferences, and selects the items with the highest predicted ratings for each user, by using the approach presented in 2.2.

Here, we describe the tasks performed by the system and how they have been implemented. The system performs three tasks:

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. *Predictions for Individual Users.* Individual predictions are calculated for each user with a User-Based Collaborative Filtering Approach presented in [26].
3. *Generation of the Group Predictions (Group modeling).* Group predictions are built by modeling the top-n items with the highest predicted rating for each user (i.e., the individual recommendations produced for each user), by averaging the ratings of the items selected for each user.

   All the steps are now described in detail.

3.2.1 Detection of the Groups (Clustering)

The first task uses the same approach previously presented for the *ModelBased* system, i.e., the k-means clustering algorithm.

3.2.2 Predictions for Individual Users

Ratings for the members of a group are predicted with a widely-used and classic User-Based Nearest Neighbor Collaborative Filtering algorithm, presented in [26]. 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. Equation 3 gives the formula used to predict the ratings:

$$p_{ui} = \overline{r}_u + \frac{\sum_{n \subset neighbors(u)} userSim(u,n) \cdot (r_{ni} - \overline{r}_n)}{\sum_{n \subset neighbors(u)} userSim(u,n)} \tag{3}$$

Values $\overline{r}_u$ and $\overline{r}_n$ represent, respectively, the mean of the ratings given by user u and user n. Similarity $userSim()$ between two users is calculated by using Pearson's correlation, a coefficient that compares the ratings of all the items rated by both the target user and the neighbor (*corated* items). Pearson's correlation between a user u and a neighbor n is given in Equation 4.

$$userSim(u,n) = \frac{\sum_{i \subset I_{un}} (r_{ui} - \overline{r}_u)(r_{ni} - \overline{r}_n)}{\sqrt{\sum_{i \subset I_{un}} (r_{ui} - \overline{r}_u)^2} \sqrt{\sum_{i \subset I_{un}} (r_{ni} - \overline{r}_n)^2}} \tag{4}$$

In the formula, I_{un} represents the set of items rated by both user u and user n. The values of the metric range from 1.0 (which indicates a complete similarity) and -1.0 (which indicates a complete dissimilarity). As highlighted in [12], negative correlations do not help increasing the prediction accuracy and can be discarded, so this approach was also followed by the task.

Note that an adapted version of this algorithm, which considers only the users in the group in order to build the predictions, was designed and implemented, but it was less accurate than the one that uses the whole user set. The analysis of this algorithm has been omitted in order to facilitate the reading of the paper.

3.2.3 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 predicted ratings for that item is calculated, in order to derive the preference of that group for the item.

3.3 System Based on the Aggregation of Individual Predictions

PredictionAggregation is a system that 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 2.3.

Here, we describe the tasks performed by the system and how they have been implemented.

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. *Predictions for Individual Users.* Individual predictions are calculated for each user with the previously presented User-Based Collaborative Filtering approach.
3. *Aggregation of the Predictions (Group modeling).* Once groups have been detected, a group model is built by aggregating all the predictions of a group.

All the tasks use the same algorithms previously presented, i.e., the k-means clustering algorithm, the User-Based Collaborative Filtering algorithm, and the *Additive Utilitarian* group modeling strategy. The difference with the *MergeRecommendation* system is on the modeling task, which considers all the predictions and not just the top-n predicted items.

4 Experimental Framework

The evaluation and comparison of the proposed systems involved more than fifty experiments and statistical tests, and this section presents the framework used to perform them. The strategy that drove our experiments is first described; then the dataset used and the preprocessing made on the data are described; after, the metrics used to make the evaluations are presented; the last part of the section presents the obtained results.

4.1 Strategy

To perform the experiments we adopted MovieLens-1M[1], a dataset widely used in the literature of recommender systems. The dataset can be easily connected to the application scenario presented in the Introduction, thinking about a company that rents movies, like Netflix. It is known that 60% of the movies rented by the company are due to recommendations[2]. A possible improvement to Netflix's approach to recommendation could be to start sending recommendation fliers to increase the amount of users to reach, instead of using only emails. Therefore, the proposed approach would be very useful in

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

[2] http://www.nytimes.com/2008/11/23/magazine/23Netflix-t.html

this real-world scenario. Even if a Netflix dataset is available, given the large amount of experiments performed to validate our proposal and the previously mentioned popularity of MovieLens in the literature, we adopted this dataset in our experimental framework.

The clusterings with k-means were created by using a testbed program, called KMlocal [16]. The program contained a variant of the k-means algorithm, called *EZ Hybrid*, developed by the authors. The k-means algorithm minimizes the *average distortion*, i.e., the mean squared distance from each data point to its nearest center. With the data used for the experiments, *EZ Hybrid* is the algorithm that returned the lowest distortion and therefore is the one used to cluster the users.

In order to identify the most performing system, a comparative analysis has been performed, by considering the RMSE values obtained by each system with different numbers of groups to detect (which correspond to the number of recommendation lists that can be produced by the system). In order to evaluate the quality of the predicted ratings for different numbers of groups, in each experiment four different clusterings of the users into 20, 50, 200, and 500 groups were created. The choice to measure the performance for different numbers of groups has been made to show how the accuracy of the systems changes as the constraint changes. 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 set of recommendations 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 algorithms when they work with groups).

RMSE was chosen as a metric to compare the algorithms because, as the organizers of the Netflix prize highlight[3], it is well-known and widely used, it allows to evaluate a system through a single number, and it emphasizes the presence of large errors (both false positives and false negatives).

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

The details of the experiments are now described.

1. *Parameter setting.* For each system, a parametric analysis for a different number of groups has been performed, in order to find the setting that allows to achieve the best accuracy for the system.
2. *Selection of the best system.* The accuracies of the systems have been compared as the number of groups changes, in order to identify the system that allows to predict the most accurate ratings.

[3] http://www.netflixprize.com/faq

4.2 Dataset and Data Preprocessing

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

For this framework, only the file `ratings.dat`, which contains the actual ratings given by 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. Each user rated at least 20 movies.

The file `ratings.dat` was preprocessed for the experimentation. Out of all the features available, just the first three were selected (i.e., *UserID*, *MovieID*, and *Rating*), since none of the presented systems uses a timestamp. The feature *UserID* was mapped into a new set of IDs between 0 and 6039, to facilitate the computation using data structures.

In order to perform statistical tests to validate the obtained results, experiments were repeated five times with a 5-fold cross-validation. In this approach, each rating available in the dataset is used four times for training and once for testing. In order to do so, the dataset is split into five subsets with a random sampling technique (each subset contains 20% of the ratings). During each run of experiments, one of the subsets becomes the test set and the rest is used for training.

4.3 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:

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

where n is the number of ratings available in the test set.

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 t-test evaluates 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 indicates the variance of the two samples, n_1 and n_2 indicate the number of values considered 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 determine the $t - value$ that indicates the result of the 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

$$t(d.f.) = t - value$$

The $t - value$ allows to derive the probability p that there is no difference between the two means. Note that when the results of a t-test are presented, the standard deviation SD of the mean is also indicated.

4.4 Experiments

The evaluation of each system is now presented. For each system, experiments to set the parameters and find the best configuration are performed. Then the accuracy of the different systems is compared.

4.4.1 Parameter Setting for the ModelBased System

Here, a description of the experiments performed to set the parameters of the system (i.e., *coratings* and n) is presented.

***coratings* parameter setting.** The *coratings* parameter, which allows the system to consider in the model only the items rated by a certain part of the group, has to be set. An experiment to evaluate a suitable value for the parameter is performed. In this experiment, parameter n is set to 10.

Fig. 4 and Table 4 show that the initial value of *coratings*, i.e., 10%, is the one that allows the system to achieve better results. This means that the higher is the value of *coratings*, the more ratings are eliminated for the model, and the harder it is for the system to predict ratings for a group.

Table 4 RMSE for the different values of *coratings*

	1 group	20 groups	50 groups	200 groups	500 groups	6040 groups
coratings=10%	1.0706	1.0402	1.0335	1.0265	1.0262	0.9120
coratings=15%	1.1086	1.0696	1.0611	1.0471	1.0428	0.9120
coratings=20%	1.1608	1.0948	1.0804	1.0672	1.0597	0.9120
coratings=25%	1.1612	1.1178	1.1011	1.0849	1.0775	0.9120
coratings=30%	1.1617	1.1417	1.1233	1.1039	1.0930	0.9120

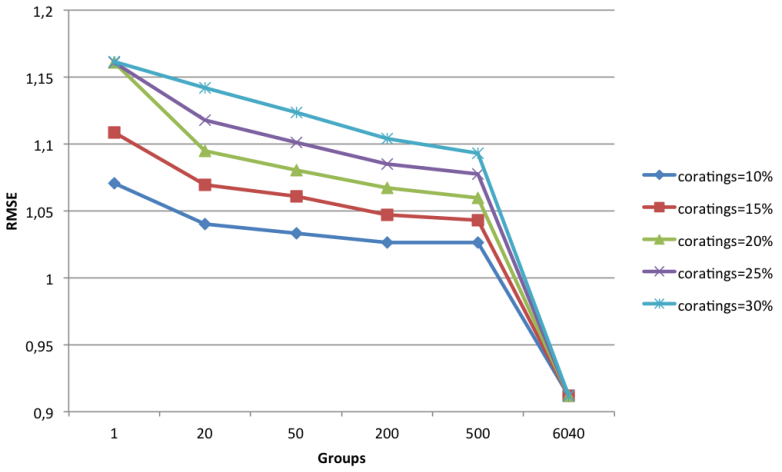

Fig. 4 RMSE for the different values of *coratings*

Independent-samples t-tests have been performed, in order to compare the results for different values of *coratings* in each clustering. All the tests returned that there is a significant difference in the values obtained with different values of the *coratings* parameter. The results of the tests that compare the performance considering 10% and 15% of the group are now presented.

Considering 1 group, there is a significant difference in the RMSE values for $coratings = 10\%$ ($M = 1.0706$, $SD = 0.00$) and $coratings = 15\%$ ($M = 1.1086$, $SD = 0.00$); $t(7.85) = 20.26$, $p = 0.0$.

For 20 groups, the difference is also significant when comparing the RMSE values for $coratings = 10\%$ ($M = 1.0402$, $SD = 0.00$) and $coratings = 15\%$ ($M = 1.0696$, $SD = 0.00$); $t(9.96) = 9.24$, $p = 0.0$.

The test performed for 50 groups returned a significant difference between $coratings = 10\%$ ($M = 1.0335$, $SD = 0.00$) and $coratings = 15\%$ ($M = 1.0611$, $SD = 0.00$); $t(7.11) = 15.24$, $p = 0.0$.

With 200 groups, the obtained results are $coratings = 10\%$ ($M = 1.0265$, $SD = 0.00$) and $coratings = 15\%$ ($M = 1.0471$, $SD = 0.00$); $t(7.88) = 14.60$, $p = 0.0$.

For 500 groups, there is a significant difference in the RMSE values for $coratings = 10\%$ ($M = 1.0263$, $SD = 0.00$) and $coratings = 15\%$ ($M = 1.0429$, $SD = 0.00$); $t(7.68) = 13.80$, $p = 0.0$.

The results suggest that lowering the *coratings* value allows the system to substantially improve the accuracy. Specifically, these results suggest that the less ratings are removed from the model, the better the algorithm predicts the ratings for a group.

Setting of the parameter *n*. In order to predict a rating for the group, the items most similar to the one currently predicted have to be selected. To

this end, the right number of neighbors has to be chosen when computing a prediction. This is done with a parameter called n, tested in this set of experiments. In this experiment, the parameter *coratings* is set to 10%.

Fig. 5 and Table 5, show the performance of the system for different values of n, i.e., considering a different number of similar items.

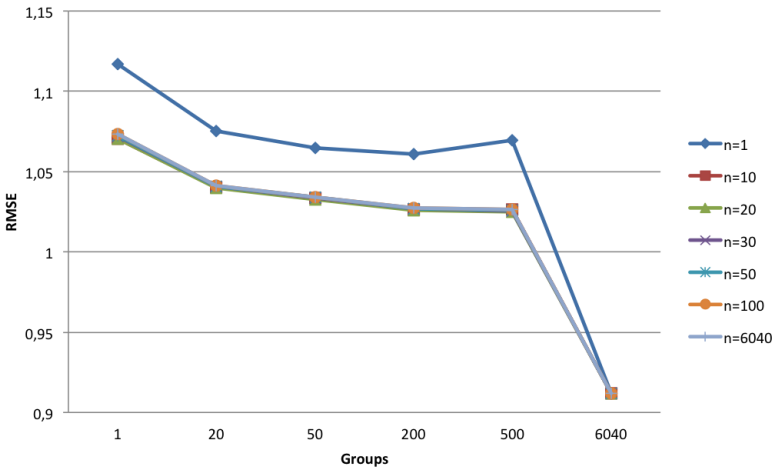

Fig. 5 RMSE for the different values of n

Table 5 RMSE for the different values of n

	1 group	**20 groups**	**50 groups**	**200 groups**	**500 groups**	**6040 groups**
n=1	1.1171	1.0752	1.0648	1.0610	1.0694	0.9120
n=10	1.0706	1.0402	1.0335	1.0265	1.0262	0.9120
n=20	1.0705	1.0398	1.0327	1.0257	1.0249	0.9120
n=50	1.0722	1.0410	1.0334	1.0267	1.0256	0.9120
n=100	1.0732	1.0412	1.0341	1.0274	1.0265	0.9120
n=6040	1.0731	1.0413	1.0342	1.0275	1.0266	0.9120

Unfortunately, the results reported in Fig. 5 are not clear and it is impossible to see the value that allowed to obtain the best results. Therefore, the figure has been zoomed in order to focus on the part between 20 and 500 groups (Fig. 6).

As the results show, there is an improvement up to $n = 20$, then results start worsening again (Fig. 6 shows very similar results for $n = 10$, represented by the red line, and $n = 30$, represented by the violet line). However, the RMSE values are very close to each other, so it is important to perform independent-samples t-tests to evaluate the difference between the results. In particular, the tests performed to compare 20 and 30 groups are now presented.

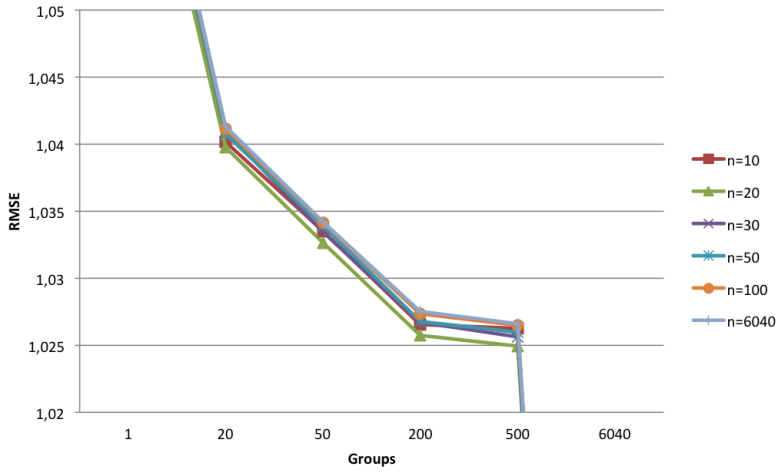

Fig. 6 Detail of the experiment to study parameter n

Considering 1 group, there is a difference in the RMSE values for $n = 20$ ($M = 1.0705$, $SD = 0.00$) and $n = 30$ ($M = 1.0722$, $SD = 0.00$); $t(9.92) = 0.87$, $p = 0.41$.

For 20 groups, there is difference for the results obtained with $n = 20$ ($M = 1.0398$, $SD = 0.00$) and $n = 30$ ($M = 1.0410$, $SD = 0.00$); $t(7.17) = 0.40$, $p = 0.70$.

The test performed for 50 groups returned a difference between $n = 20$ ($M = 1.0334$, $SD = 0.00$) and $n = 30$ ($M = 1.0339$, $SD = 0.00$); $t(7.36) = 0.49$, $p = 0.63$.

With 200 groups, there is a difference between the RMSE values obtained with $n = 20$ ($M = 1.0267$, $SD = 0.00$) and $n = 30$ ($M = 1.0268$, $SD = 0.00$); $t(7.31) = 0.74$, $p = 0.48$.

For 500 groups, the test returned a difference between $n = 20$ ($M = 1.0249$, $SD = 0.00$) and $n = 30$ ($M = 1.0256$, $SD = 0.00$); $t(7.94) = 0.67$, $p = 0.52$.

As it can be noticed, the results of the t-tests show that there is not enough confidence to reject the null hipotesys that the values obtained for $n = 20$ and $n = 30$ are different. However, the results obtained with $n = 20$ are always better in terms of RMSE and the t-tests returned that the probability that there is a difference for $n = 20$ ranges between 30% and 59%. Therefore, the value of n used to select the items similar to the one considered is 20.

4.4.2 Parameter Setting for the MergeRecommendations System

Here, a description of the experiments performed to estimate the two parameters used by the algorithm (i.e., *neighbors* and n) is presented.

Selection of the number of *neighbors*. In order to predict a rating for a

user, the users most similar to the one currently considered have to be selected. In order to do so, the right number of neighbors has to be chosen when computing a prediction. This is done with a parameter called *neighbors*, tested in this set of experiments.

Since we have to evaluate the number of neighbors for an algorithm that predicts individual ratings, this evaluation is done out of the group recommendation context. In other words, the RMSE values of the individual predictions for different values of *neighbors* are presented.

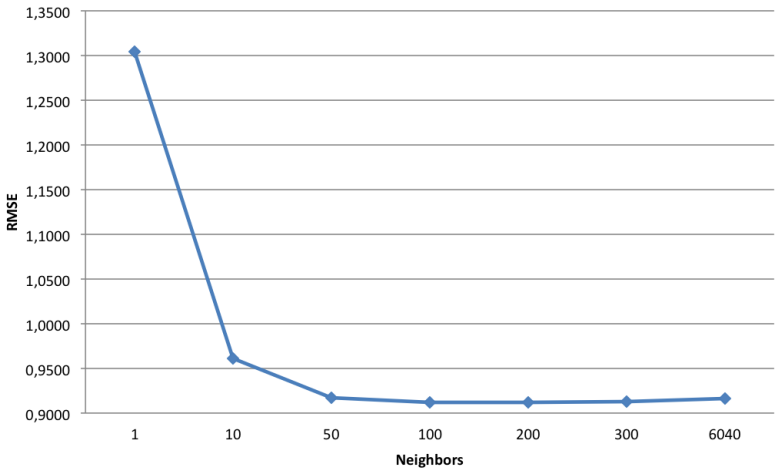

Fig. 7 RMSE values for increasing number of *neighbors*

Table 6 RMSE for increasing number of *neighbors*

	6040 groups
neighbors=1	1.3046
neighbors=10	0.9611
neighbors=50	0.9167
neighbors=100	0.9118
neighbors=200	0.9120
neighbors=300	0.9128
neighbors=6040	0.9160

Fig. 7 and Table 6 show the RMSE values for increasing values of *neighbors*. As highlighted in [10], this is the common way to choose the value. Moreover, our results reflect the trend described by the authors, i.e., for low values of the parameter, a great improvement can be noticed. As expected, RMSE takes the form of a convex function (Fig. 8 shows a detail of Fig. 7), which indicates

that after a certain value improvement stops. In these experiments that value is 100.

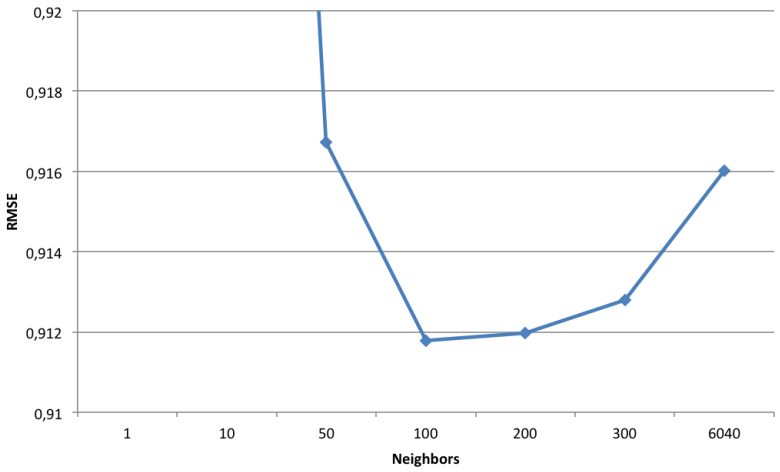

Fig. 8 RMSE takes the form of a convex function.

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

There is a significant difference in the RMSE values for 1 neighbor ($M = 1.3046$, $SD = 0.00$) and 100 neighbors ($M = 0.9116$, $SD = 0.00$); $t(7.59) = 450.02$, $p = 0.00$.

There is a also a significant difference in the RMSE values for 10 neighbors ($M = 0.9611$, $SD = 0.00$) and 100 neighbors ($M = 0.9118$, $SD = 0.00$); $t(7.41) = 54.44$, $p = 0.00$.

A significant difference is also present in the RMSE values for 50 neighbors ($M = 0.9167$, $SD = 0.00$) and 100 neighbors ($M = 0.9118$, $SD = 0.00$); $t(7.97) = 6.02$, $p = 0.00$.

The RMSE values present a difference for 100 neighbors ($M = 0.9118$, $SD = 0.00$) and 200 neighbors ($M = 0.9120$, $SD = 0.00$); $t(7.99) = 0.24$, $p = 0.82$.

There is a also a difference in the RMSE values for 100 neighbors ($M = 0.9118$, $SD = 0.00$) and 300 neighbors ($M = 0.9128$, $SD = 0.00$); $t(7.97) = 1.06$, $p = 0.33$.

There is a significant difference in the RMSE values for 100 neighbors ($M = 0.9118$, $SD = 0.00$) and 6040 neighbors ($M = 0.916$, $SD = 0.03$); $t(7.99) = 1.27$, $p = 0.24$.

As it can be noticed, for values of *neighbors* higher than 100, the probability that there is a difference between the values obtained for 100 and 200

neighbors and 100 and 300 neighbors is between 18% and 67%. In particular, there seems to be no difference between choosing 100 and 200 neighbors. Since $neighbors = 100$ returned the best results and it is clearly faster to compute predictions considering only 100 neighbors instead of 200, this is the value chosen for the algorithm.

Choice of the top-n items. *MergeRecommendations* works by combining recommendations made for individual users, in the form of the items that received the highest predicted ratings (top-n items). This set of experiments allows us to evaluate how big n should be, i.e., how many items should be selected from the predictions calculated for every user.

Fig. 9 and Table 7 show that selecting the top-5 out of all the predicted ratings allows to achieve the best results.

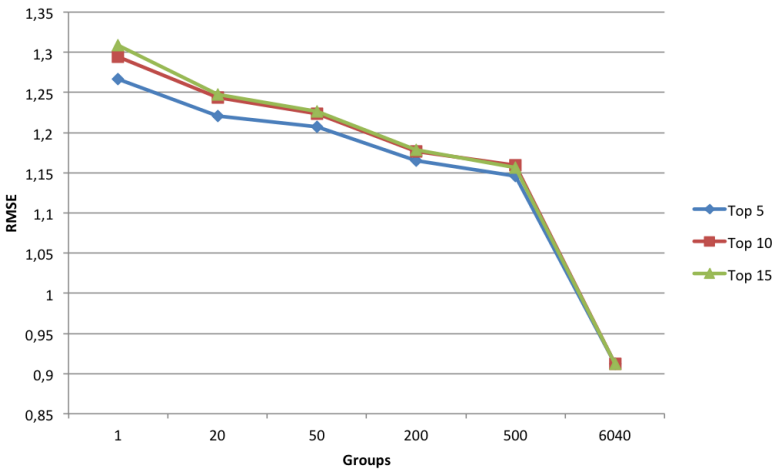

Fig. 9 RMSE values for increasing values of n

Table 7 RMSE for the different values of the top n ratings

	1 group	**20 groups**	**50 groups**	**200 groups**	**500 groups**	**6040 groups**
Top 5	1.2667	1.2207	1.2075	1.1653	1.1461	0.9120
Top 10	1.2947	1.2437	1.2235	1.1762	1.1592	0.9120
Top 15	1.3092	1.2473	1.2262	1.1788	1.1562	0.9120

Independent-samples t-tests have been performed, in order to evaluate if there is a significant difference between the values obtained for the different values of n and different numbers of groups. Such a difference exists and the results of the tests that compare $n = 5$ and $n = 10$ (i.e., the values that obtained the most similar results) are now presented.

Considering 1 group, there is a significant difference between $n = 5$ ($M = 1.2667$, $SD = 0.00$) and $n = 10$ ($M = 1.2947$, $SD = 0.00$); $t(7.74) = 3.39$, $p = 0.01$.

For 20 groups, there is a difference between $n = 5$ ($M = 1.2208$, $SD = 0.00$) and $n = 10$ ($M = 1.2437$, $SD = 0.00$); $t(7.99) = 1.46$, $p = 0.18$.

When 50 groups are considered, there is also a difference between $n = 5$ ($M = 1.2076$, $SD = 0.00$) and $n = 10$ ($M = 1.2235$, $SD = 0.00$); $t(7.98) = 0.65$, $p = 0.53$.

For 200 groups, there is also a difference between $n = 5$ ($M = 1.1653$, $SD = 0.00$) and $n = 10$ ($M = 1.1762$, $SD = 0.00$); $t(7.99) = 0.54$, $p = 0.60$.

With 500 groups, there is a difference between $n = 5$ ($M = 1.1461$, $SD = 0.00$) and $n = 10$ ($M = 1.1591$, $SD = 0.00$); $t(7.45) = 0.65$, $p = 0.54$.

Results show that when the number of groups increases, the significance of the difference between the values decreases. However, since for $n = 5$ the results are always lower and the highest probability that the values are not different is 40%, the value was chosen for the algorithm.

4.4.3 Parameter Setting for the PredictionAggregation System

This system automatically detects groups according to the constraints imposed by the system, predicts individual ratings, and models the groups.

Since the algorithm that predicts individual ratings is the same used by *MergeRecommendations* and it was previously tested, no additional experiments have to be performed. The system was run with the previously tested value of the *neighbors* parameter, i.e., $neighbors = 100$ and results are shown in Fig. 10 and Table 8.

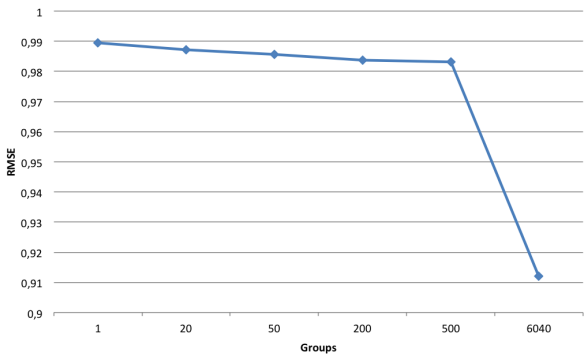

Fig. 10 RMSE values of *PredictionAggregation* for a different number of groups

Table 8 RMSE values of *PredictionsAggregation*

	1 group	20 groups	50 groups	200 groups	500 groups	6040 groups
RMSE	0.9895	0.9872	0.9857	0.9837	0.9832	0.9120

4.4.4 Selection of the best system

Fig. 11 and Table 9 report the results obtained by each system with its best configuration.

An aspect not previously deepened in the previous experiments in order to focus on the specific aspects that were tested, is that for all the systems, as the number of groups grows, the accuracy improves. So as the number of groups increases, the RMSE values get lower. In other words, as expected, a system can have a better performance when more recommendations can be produced.

As it can be noticed, the three approaches to produce group recommendations are clearly separated in the results.

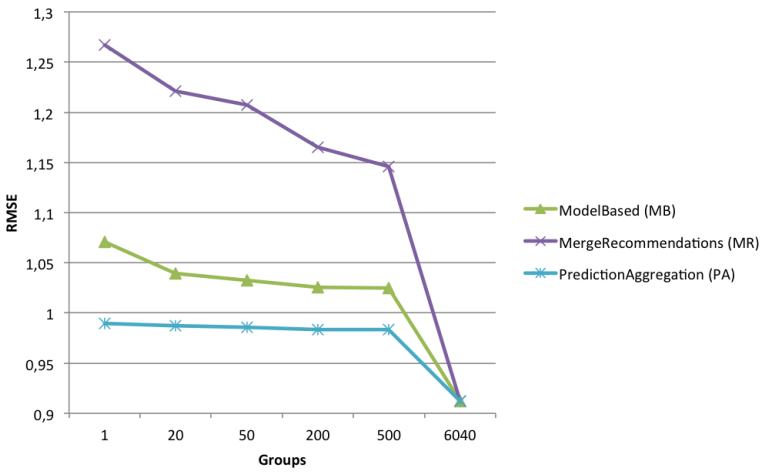

Fig. 11 RMSE values obtained by the best system

Table 9 RMSE obtained by the each system

	1 group	20 groups	50 groups	200 groups	500 groups	6040 groups
MB	1.0705	1.0398	1.0327	1.0257	1.0249	0.9120
MR	1.2667	1.2207	1.2075	1.1653	1.1461	0.9120
PA	0.9895	0.9872	0.9857	0.9837	0.9832	0.9120

In fact, the system that merges individual recommendations (implemented in the *MergeRecommendations* system) achieves the worst results. This is the sign that with large and automatically detected groups, if user preferences are expressed just with a small subset of items (in this case five), a group recommendation algorithm is not able to properly satisfy users.

The approach based on a group model (i.e., *ModelBased*) lays in the middle.

At the bottom of the figure, achieving the best results, there is the system that merges individual preferences (i.e., the *PredictionAggregation* system). This means that predicting the ratings for each user and considering all the predictions in the group models leads to great improvements in the quality of the predicted results.

Independent-samples t-tests, performed to compare the results, confirm that there is a significant difference between the RMSE values obtained by *PredictionsAggregation* and the ones obtained with the other systems.

The results obtained by comparing the two closest systems in terms of their performance (i.e., *ModelBased* and *PredictionAggregation*) are now presented.

Considering 1 group, there is a significant difference in the RMSE values for *ModelBased* ($M = 1.0705$, $SD = 0.00$) and *PredictionAggregation* ($M = 0.9895$, $SD = 0.00$); $t(4.23) = 65.14$, $p = 0.0$.

For 20 groups, the difference is also significant when comparing the RMSE values for *ModelBased* ($M = 1.0398$, $SD = 0.00$) and *PredictionAggregation* ($M = 0.9872$, $SD = 0.00$); $t(4.88) = 39.68$, $p = 0.0$.

The test performed for 50 groups returned a significant difference between *ModelBased* ($M = 1.0327$, $SD = 0.00$) and *PredictionAggregation* ($M = 0.9857$, $SD = 0.00$); $t(4.77) = 39.68$, $p = 0.0$.

With 200 groups, the obtained results are *ModelBased* ($M = 1.0257$, $SD = 0.00$) and *PredictionAggregation* ($M = 0.9837$, $SD = 0.00$); $t(6.24) = 40.46$, $p = 0.0$.

For 500 groups, there is a significant difference in the RMSE values for *ModelBased* ($M = 1.0249$, $SD = 0.00$) and *PredictionAggregation* ($M = 0.9832$, $SD = 0.00$); $t(7.07) = 50.50$, $p = 0.0$.

These results suggest that building predictions for individual users instead of building them for each group, leads to significant improvements in the accuracy of a group recommender system that detects groups.

4.5 Results Discussion

Results show that building individual predictions for each user and then aggregating them into a group prediction (as the *PredictionAggregation* system does) leads to a better performance with respect to the approaches that build specific predictions for a group (i.e., the *ModelBased* system), or merge individual recommendations (i.e., the *MergeRecommendations* system).

This is due to the fact that with large groups (which is usually the case when the number of recommendations is limited), the last two approaches tend to lose precision when recommendations are built. In fact, by using a model-based approach, specific predictions for a group are built by using aggregated preferences, which cannot reflect the preferences of every user in a group. Regarding the approach that produces group recommendations by merging individual recommendations, the performance is affected, because each user of a group is only represented by a limited set of items (the top-n items with the highest predicted value). Therefore, these results show that discarding from a user model too many ratings (as the approach that considers only the individual recommendations does), leads to a significant worsening of the performance, even if the predictions are built in the same way as the most performing system.

In conclusion, the accuracy of a group recommender system that automatically detects groups by clustering is affected by the rating prediction task both in the way ratings are predicted, and by the amount of predicted ratings that are considered to build the group recommendations.

5 Conclusions

This paper explored the rating prediction task in a group recommendation scenario in which groups are automatically detected by cluster users.

In the first part of the work, the existing approaches to predict the ratings in a group recommender system have been presented. Then, we proposed three recommender systems that automatically detect groups and implement the different ways to predict the ratings. The study continued with a deep evaluation of all the proposed systems. All the systems have then been compared and experimental results showed that the system that combines individual preferences is the one that works best with automatically detected groups.

In conclusion, the proposed systems and the extensive analysis performed on them allowed to achieve interesting contributions, now recapped.

- A system that uses all the predictions built for the individual users can achieve a better performance with respect to the approaches that build specific predictions for a group or use only the items with the highest predicted ratings.
- The accuracy improves as the number of groups grows. An analysis of the behavior of a system for different numbers of groups allowed us to explore the trade-off between the number of recommendation lists produced and the accuracy of the system.

Recent studies [18, 19] showed that behavioral data mining is an efficient tool to create connections between the users of a social network. Future work will study our group recommendation approach in contexts in which the behavior of the users in a social network can be exploited, in order to cluster the users based on the behavioral features.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005)
2. Amatriain, X., Jaimes, A., Oliver, N., Pujol, J.M.: Data mining methods for recommender systems. In: F. Ricci, L. Rokach, B. Shapira, P.B. Kantor (eds.) Recommender Systems Handbook, pp. 39–71. Springer, Boston, MA (2011)
3. 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, pp. 1–20. Springer Berlin Heidelberg (2011)
4. Boratto, L., Carta, S.: Exploring the ratings prediction task in a group recommender system that automatically detects groups. In: IMMM 2013, The Third International Conference on Advances in Information Mining and Management, pp. 36–43 (2013)
5. Boratto, L., Carta, S.: Modeling the preferences of a group of users detected by clustering: A group recommendation case-study. In: Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14), WIMS '14, pp. 16:1–16:7. ACM, New York, NY, USA (2014). DOI 10.1145/2611040.2611073. URL http://doi.acm.org/10.1145/2611040.2611073
6. 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 Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 03, WI-IAT '09, pp. 547–550. IEEE Computer Society, Washington, DC, USA (2009). DOI 10.1109/WI-IAT.2009.346. URL http://dx.doi.org/10.1109/WI-IAT.2009.346
7. Boratto, L., Carta, S., Satta, M.: Groups identification and individual recommendations in group recommendation algorithms. In: J. Picault, D. Kostadinov, P. Castells, A. Jaimes (eds.) Practical Use of Recommender Systems, Algorithms and Technologies 2010, *CEUR Workshop Proceedings*, vol. 676 (2010). URL http://ceur-ws.org/Vol-676/paper4.pdf
8. 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)
9. Crossen, A., Budzik, J., Hammond, K.J.: Flytrap: intelligent group music recommendation. In: Proceedings of the 7th international conference on Intelligent user interfaces, IUI '02, pp. 184–185. ACM, New York, NY, USA (2002)
10. Desrosiers, C., Karypis, G.: A comprehensive survey of neighborhood-based recommendation methods. In: Recommender Systems Handbook, pp. 107–144. Springer, Berlin (2011)
11. Goren-Bar, D., Glinansky, O.: Fit-recommend ing tv programs to family members. Computers & Graphics **28**(2), 149–156 (2004)
12. 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 (1999)
13. Jameson, A.: More than the sum of its members: challenges for group recommender systems. In: Proceedings of the working conference on Advanced visual interfaces, pp. 48–54. ACM Press (2004)
14. Jameson, A., Smyth, B.: Recommendation to groups. In: The Adaptive Web, Methods and Strategies of Web Personalization, pp. 596–627. Springer (2007)
15. 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)
16. 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)
17. 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)

18. 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)
19. 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)
20. Masthoff, J.: Group recommender systems: Combining individual models. In: Recommender Systems Handbook, pp. 677–702. Springer (2011)
21. McCarthy, J.: 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 (2002)
22. McCarthy, J.F., Anagnost, T.D.: Musicfx: An arbiter of group preferences for computer supported collaborative workouts. In: S.E. Poltrock, J. Grudin (eds.) CSCW '98, Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work, Seattle, WA, USA, November 14-18, 1998, pp. 363–372. ACM (1998)
23. McCarthy, K., Salamó, M., Coyle, L., McGinty, L., Smyth, B., Nixon, P.: Cats: A synchronous approach to collaborative group recommendation. In: G. Sutcliffe, R. Goebel (eds.) Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference, Melbourne Beach, Florida, USA, May 11-13, 2006, pp. 86–91. AAAI Press (2006)
24. O'Connor, M., Cosley, D., Konstan, J.A., Riedl, J.: Polylens: A recommender system for groups of user. In: Proceedings of the Seventh European Conference on Computer Supported Cooperative Work, pp. 199–218. Kluwer (2001)
25. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Recommender Systems Handbook, pp. 1–35. Springer, Berlin (2011)
26. Schafer, J.B., Frankowski, D., Herlocker, J.L., Sen, S.: Collaborative filtering recommender systems. In: The Adaptive Web, Methods and Strategies of Web Personalization, pp. 291–324. Springer (2007)
27. Stettinger, M., Felfernig, A.: Choicla: Intelligent decision support for groups of users in context of personnel decisions. In: Proceedings of the ACM RecSys'2014 IntRS Workshop, pp. 28–32 (2014)
28. Yu, Z., Zhou, X., Hao, Y., Gu, J.: Tv program recommendation for multiple viewers based on user profile merging. User Modeling and User-Adapted Interaction **16**(1), 63–82 (2006)
29. Zhiwen, Y., Xingshe, Z., Daqing, Z.: An adaptive in-vehicle multimedia recommender for group users. In: Proceedings of the 61st Semiannual Vehicular Technology Conference, vol. 5, pp. 2800–2804 (2005)