Exploring the Ratings Prediction Task in a Group Recommender System that Automatically Detects Groups

Ludovico Boratto and Salvatore Carta
Dip.to di Matematica e Informatica
Università di Cagliari
Via Ospedale 72
09124 Cagliari, Italy
Email: {ludovico.boratto, salvatore}@unica.it

***Abstract*—Recommender systems produce content for users, by suggesting items that users might like. Predicting the ratings is a key task in a recommender system. This is especially true in a system that works with groups, because ratings might be predicted for each user or for the groups. The approach chosen to predict the ratings changes the architecture of the system and what information is used to build the predictions. This paper studies approaches to predict the ratings in a group recommendation scenario that automatically detects groups. Experimental results confirm that the approach to predict the ratings strongly influences the performances 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 quality of the recommendations.**

***Keywords*—Group Recommendation; Clustering; Ratings Prediction.**

I. 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" [1]. As highlighted in [1], [2], the prediction task is the core recommendation computation.

Group recommendation is designed for contexts in which more than a person is involved in the recommendation process [3]. A scenario in which group recommendation is useful is when the recommendations that can be built are limited.

> *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, these systems add the complexity of defining groups, in order to respect the constraint on the number of recommendations that can be produced. In literature, no system can automatically adapt to such constraints imposed by the system.

According to Jameson and Smyth [3], a group recommender system can use three approaches to predict the ratings: (i) construct group models and predict the ratings for each group using the model, (ii) predict the ratings for each user and merge only the individual recommendations into a group preference, or (iii) aggregate all the predictions built for each user into a group preference. It can be noticed that the ratings prediction task takes a central role also in a group recommender system, since ratings can be predicted for each user or for a group. According to the approach chosen to predict the ratings, the architecture of the system changes and the prediction task takes a different input (i.e., a group model or the individual preferences of each user) and produces a 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.

This paper explores the ratings prediction task in the previously mentioned scenario, in order to identify the best approach to predict the ratings for a group that has been automatically detected. Three recommender systems have been developed, to produce the predictions according to the previously mentioned approaches. The scientific contributions coming from this paper are the following: (i) the prediction task is explored for the first time in a scenario in which groups are automatically detected; (ii) the three approaches to build the predictions in a group recommender systems are directly compared for the first time in literature and (iii) the trade-off between the number of detected groups and the accuracy of each system is explored, in order to evaluate how the number of groups affects the performances of a system.

The paper is structured as follows: Section II describes the ratings prediction approaches in group recommender systems; Section III presents the three group recommender systems that automatically detect groups, developed to use the three approaches to build the predictions; Section IV illustrates the experiments on the systems; Section V contains conclusions.

II. RATINGS PREDICTION APPROACHES IN GROUP RECOMMENDATION

According to [3], group preferences can be predicted using three approaches: (i) generation of a group model that

 ISBN: 978-1-61208-311-7

combines individual preferences, used to build predictions for the group, (ii) merging of recommendations built for each user, or (iii) aggregation of the predictions built for each user.

This section describes each approach in detail.

A. Construction of Group Preference Models

This approach builds a group model using the preferences of the users and predicts a rating for the items not rated by the group using the model. The two performed tasks are:

1) Construct a model M_g for a group g, that 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 shown in Fig. 1 (the prediction task is highlighted in the figure). In order to build the predictions, the system has to produce a model with the preferences of the group (TASK 1). The task receives as input the ratings for the items 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).

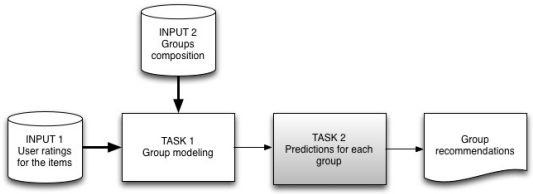

Fig. 1. System that builds predictions using group models

This approach is used by *Let's Browse* [4], *In-Vehicle Multimedia Recommender* [5], *TV4M* [6], *INTRIGUE* [7], and *Travel Decision Forum* [8].

B. Merging of Individual Recommendations

The approach presents to a group a set of items, that 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, predict a rating p_{ui}.
   - Select the set C_i of items with the highest predicted ratings p_{ui}.
2) Model the group preferences by producing $\bigcup_i C_i$, the union of the sets of items with the highest predicted ratings.

The architecture of a system that uses this approach is shown in Fig. 2. The system uses the ratings for the items evaluated by each user (INPUT 1) to predict the ratings for each user (TASK 1). The output produced by the task (the top-n predictions, used as recommendations for a user), are given as input to the task that merges the recommendations (TASK 2), along with the composition of the groups (INPUT 2).

This approach is not widely used in literature. The main relevant work that embraces this approach is *PolyLens* [9].

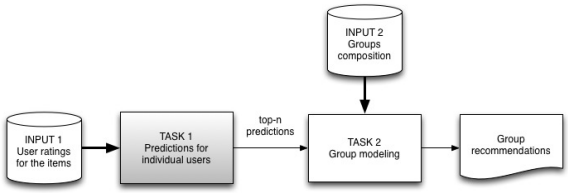

Fig. 2. System that merges the recommendations

C. Aggregation of Individual Predictions

This approach predicts individual preferences for the items not rated by each user, aggregates individual preferences and derives a group preference. The approach works as follows.

- For each item i:
  1) For each member u of the group g that did not rate i, predict a rating p_{ui}.
  2) 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 ratings for the item evaluated by each user (INPUT 1) are used to predict the ratings for the missing items for each user (TASK 1). The output produced by the task (i.e., 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).

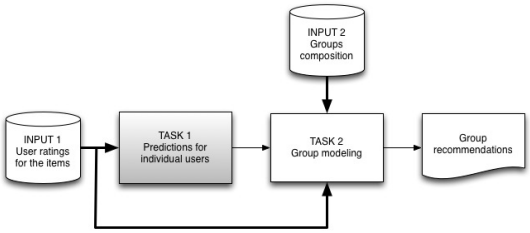

Fig. 3. System that aggregates predictions

This approach is used by *PolyLens* [9] and in [10], [11].

III. Predicting Ratings for Automatically Detected Groups

As mentioned in the Introduction, no group recommendation approach is able to detect groups in order to adapt to constraints on the number of recommendations produced. This section presents three group recommender systems able to automatically detect groups. Each system will implement one of the three approaches to predict the ratings previously described. The tasks that do not predict ratings will be implemented in the same way in all the systems, in order to evaluate how each approach to predict the ratings affects the performances of a group recommender system.

 ISBN: 978-1-61208-311-7

A. System Based on Group Models Construction

ModelBased is a group recommender system that detects groups of similar users, models each group using the preferences of its members and predicts group preferences using the model, according to the approach presented in Section II-A. The tasks performed by the system are the following.

1) *Detection of the groups.* Using the preferences of each user, groups of users with similar preferences are detected with the k-means clustering algorithm [12].
2) *Group modeling.* Once groups have been detected, a group model is built for each group g, using the *Additive Utilitarian* modeling strategy. For each group, a rating is calculated for a subset of items.
3) *Prediction of the Ratings Using a Group Model.* Group ratings are predicted for the items not modeled by the previous task, using the model that contains its preferences with an Item-based approach [13].

Detection of the groups. The set of users has first to be partitioned into a number of groups equal to the number of recommendations. Since in our application scenario groups do not exist, unsupervised classification (*clustering*) is necessary. Users are clustered considering the ratings expressed for the evaluated items. It was recently highlighted in [14] that the k-means clustering algorithm [12] is by far the most used clustering algorithm in recommender systems.

This task detects groups by clustering users with the k-means clustering algorithm. The output of the task is a partitioning of the users in groups (clusters), such that users with similar ratings for the same items are in the same group and can receive the same recommendations.

Group modeling. To create a model that represents the preferences of a group, the *Additive Utilitarian* group modeling strategy [15] is adopted. The strategy sums individual ratings for each item and produces a list of the group ratings (the higher the sum is, the earlier the item appears in the list). The ranked group list of items is exactly the same that would be produced when averaging the individual ratings, so this strategy is also called 'Average strategy'.

The choice to use this strategy to create the model and produce ratings using an average was made for two main reasons: (i) since the considered scenario deals with a limited number of recommendations, the system works with large groups. Therefore, an average, that is a single value that is meant to typify a set of different values, is best way to put together the ratings in this context; (ii) 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.

After a group has been modeled, a rating 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 rating cannot be considered representative of the preferences of the group as a whole. 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 rating in the model.

Prediction of the Ratings Using a Group Model. In the group models previously created, for a subset of items there is no preference. In order to predict these ratings, an Item-Based Nearest Neighbor Collaborative Filtering algorithm presented in [13], that builds the predictions using the model, is adopted. The choice of using an Item-Based approach is because the algorithm deals with group models. Since groups might be very large, a group model might put together a lot preferences and it would not be significant to make a prediction with a User-based approach that would look for "similar groups". In fact, considering an example with 6000 users and 10 groups, if groups were homogeneous, there would be around 600 users per group. If a User-Based approach was used, when looking for neighbors the algorithm would look for a two similar models, that each contain a synthesis of the preferences of 600 users. This type of similarity would not be accurate enough to make predictions.

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

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

According to Schafer et al., [13], some authors do not consider all the items in the model (i.e., $ratedItems(g)$), but just the top n correlations. In order to reduce the computational complexity of the algorithm and select the most meaningful correlations, this is the approach used for this task. In order to compute similarity $itemSim(i,j)$ between two items, adjusted-cosine similarity is used. The metric is believed to be the most accurate when calculating similarities between items [13]. It is computed 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 and value $\overline{r}_u$ represents the mean of the ratings expressed by user u).

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

B. 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, using the approach presented in Section II-B. Here we describe the tasks performed by the system and how they have been implemented.

1) *Detection of the groups.* Considering the individual preferences, groups of similar users are detected with 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 [13].
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, by averaging the ratings of the items selected for each user.

 ISBN: 978-1-61208-311-7

Detection of the Groups. The first task uses the approach previously presented, i.e., the k-means algorithm.

Prediction of the Missing Ratings. Ratings for the members of a group are predicted with a classic User-Based Nearest Neighbor Collaborative Filtering algorithm, presented in [13]. The algorithm predicts a rating p_{ui} for each item i that was not evaluated by a user u, 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:

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

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' correlation between a user u and a neighbor n is given in Equation (4) (I_{un} is the set of items rated by both u and n).

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

The metric ranges between 1.0 (complete similarity) and -1.0 (complete dissimilarity). Negative values do not increase the prediction accuracy [16], so they are discarded by the task.

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 merging the top-n items of each users with a union of the ratings. 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 (*Additive Utilitarian* strategy), in order to derive the preference of that group for the item.

C. System Based on the Aggregation of Individual Predictions

PredictionAggregation is a group recommender 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 Section II-C. Here we describe the tasks performed by the system.

1) *Detection of the groups.* Using individual preferences, groups are detected through the k-means algorithm.
2) *Predictions for Individual Users.* Predictions are built with the previously described User-Based 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* modeling strategy. The difference with the *MergeRecommendation* system is on the modeling task, that considers all the predictions and not just the top-n predicted items.

IV. Experimental Framework

This section presents the framework built for the experiments.

A. Experimental Setup

To conduct the experiments, we adopted MovieLens-1M, a dataset widely used in literature.

The clusterings with k-means were created using a testbed program called KMlocal [17], that contained a variant of the k-means algorithm, called *EZ Hybrid*. The k-means algorithm minimizes the *average distortion*, i.e., the mean squared distance from each point to its nearest center. With the dataset used, *EZ Hybrid* is the algorithm that returned the lowest distortion and is the one used to cluster the users.

An analysis has been performed, by comparing the RMSE values obtained by each system considering different numbers of groups to detect. The choice of measuring the performances for different numbers of groups has been made to show how the quality of the systems change as the constraint changes. In each experiment, four different clusterings in 20, 50, 200 and 500 groups were created. Moreover, we compared the results obtained with the four clusterings with the results obtained considering a single group with all the users (i.e., predictions are calculated considering the preferences of all the users), and the results obtained by the system that calculates predictions for each user.

RMSE was chosen as a metric to compare the algorithms because, as the organizers of the Netflix prize highlight [19], it is widely used, allows to evaluate a system through a single number and emphasizes the presence of large errors.

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 conducted. In order to make the tests, a 5-fold cross-validation was preformed.

The details of the experiments are described below.

1) *Parameters setting.* For each system, a parametric analysis has been conducted, in order to find the setting that allows to achieve the best performances.
2) *Selection of the best system.* The performances of the systems have been compared, in order to identify the one that allows to predict the most accurate ratings.

B. Dataset and Data Preprocessing

The dataset used, i.e., MovieLens-1M, is composed of 1 million ratings, expressed by 6040 users for 3900 movies. This framework uses only the file `ratings.dat`, that contains the ratings given by users. The file contains four features: *UserID*, that contains user IDs in a range between 1 and 6040, *MovieID*, that contains movie IDs in a range between 0 and 3952, *Rating*, that contains values in a scale between 1 and 5 and *Timestamp*, that contains a timestamp of when a user rated an item. The file was preprocessed for the experimentation, by mapping the feature *UserID* in a new set of IDs between 0 and 6039, to facilitate the computation using data structures. In order to conduct the cross-validation, the dataset was split into five subsets with a random sampling technique (each subset contains 20% of the ratings).

 ISBN: 978-1-61208-311-7

C. Metrics

The quality of the predicted ratings was measured through the Root Mean Squared Error (RMSE). The metric compares each rating r_{ui}, expressed by a user u for an item i in the test set, 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 conducted. These tests allow to reject the null hypothesis that two values are statistically the same. So, a two-tailed test will test if an RMSE value is significantly greater or significantly smaller than another RMSE value. Since each experiment was conducted five times, the means M_i and M_j of the RMSE values obtained by two systems i and j are used to compare the systems 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 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 results of the test), can be obtained in a standard table of significance as

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

The $t - value$ derives the probability p that there is no difference between the two means. Along with the result of a t-test, the standard deviation SD of the mean is presented.

D. Experiments

For each system, experiments to set the parameters and find the best configuration are conducted. Then, the performances of the different systems are compared.

1) Setting the Parameters of ModelBased: Here we describe the experiments conducted to set the two parameters of the system (i.e., $coratings$ and n).

***coratings* parameter setting.** The $coratings$ parameter allows to consider in the model only the items rated by a certain part of the group. An experiment to evaluate a suitable value for the parameter is conducted. In this experiment, parameter n is set to 10. Fig. 4 and the underlying table, show that the initial value of $coratings$, i.e., 10%, is the one that allows to achieve better results. This means that the higher is the value of $coratings$, the more ratings are eliminated for the model. So, it is harder to predict the ratings.

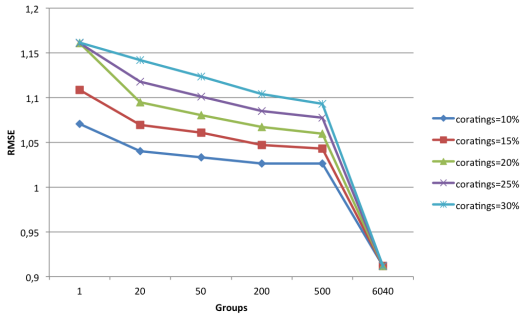

Fig. 4. RMSE for the different values of *coratings*

TABLE I. 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

Independent-samples t-tests have been conducted, 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 obtained to compare the results obtained considering 10% and 15% of the group are now presented. Considering 1 group, there is a significant difference in the RMSE values for $coratings = 10\%$ ($M = 1.070556$, $SD = 0.00$) and $coratings = 15\%$ ($M = 1.108634$, $SD = 0.00$); $t(7.85) = 20.26$, $p = 0.0$. For 20 groups, the difference is also significant when comparing the RMSE values for $coratings = 10\%$ ($M = 1.04019$, $SD = 0.00$) and $coratings = 15\%$ ($M = 1.069618$, $SD = 0.00$); $t(9.96) = 9.24$, $p = 0.0$. The test conducted for 50 groups returned a significant difference between $coratings = 10\%$ ($M = 1.033476$, $SD = 0.00$) and $coratings = 15\%$ ($M = 1.06113$, $SD = 0.00$); $t(7.11) = 15.24$, $p = 0.0$. With 200 groups, the obtained results are $coratings = 10\%$ ($M = 1.026542$, $SD = 0.00$) and $coratings = 15\%$ ($M = 1.047102$, $SD = 0.00$); $t(7.88) = 14.60$, $p = 0.0$. For 500 groups, there is a significant difference in the RMSE values for $coratings = 10\%$ ($M = 1.026246$, $SD = 0.00$) and $coratings = 15\%$ ($M = 1.042848$, $SD = 0.00$); $t(7.68) = 13.80$, $p = 0.0$. The results suggest that lowering the $coratings$ value allows to substantially improve the results. Specifically, these results suggest that the less ratings are removed from the model, the better the algorithm predicts the ratings for a group.

Setting parameter *n*. To predict a rating for the group, the items most similar to the one currently predicted are selected. In order to choose the number of neighbors, a parameter n has to be set. Parameter $coratings$ is set to 10%. Fig. 5 and the underlying table, show the performances of the system for different values of n, i.e., considering a different number of similar items. In the results reported in the figure it is hard to see the value that allows to obtain the best results, so Fig. 6 (that focuses on the part between 20 and 500 groups) shows an improvement up to $n = 20$, then results worsen again.

 ISBN: 978-1-61208-311-7

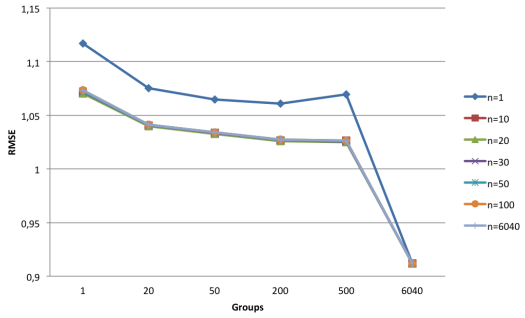

Fig. 5. RMSE for the different values of *n*

TABLE II. 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

RMSE values are very close; so, it is important to conduct independent-samples t-tests to evaluate the difference between the results. In particular, the tests conducted to compare 20 and 30 groups are reported. For 1 group, there is a difference in the RMSE values for $n = 20$ ($M = 1.070534$, $SD = 0.00$) and $n = 30$ ($M = 1.07217$, $SD = 0.00$); $t(9.92) = 0.87$, $p = 0.41$. Considering 20 groups, there is a difference in the results obtained with $n = 20$ ($M = 1.039798$, $SD = 0.00$) and $n = 30$ ($M = 1.040968$, $SD = 0.00$); $t(7.17) = 0.40$, $p = 0.70$. The test conducted for 50 groups returned a difference between $n = 20$ ($M = 1.03344$, $SD = 0.00$) and $n = 30$ ($M = 1.033898$, $SD = 0.00$); $t(7.36) = 0.49$, $p = 0.63$. With 200 groups, there is a difference between the RMSE values obtained with $n = 20$ ($M = 1.026698$, $SD = 0.00$) and $n = 30$ ($M = 1.026764$, $SD = 0.00$); $t(7.31) = 0.74$, $p = 0.48$. For 500 groups, the test returned a difference between $n = 20$ ($M = 1.02493$, $SD = 0.00$) and $n = 30$ ($M = 1.025626$, $SD = 0.00$); $t(7.94) = 0.67$, $p = 0.52$. 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%. So the value of n used is 20.

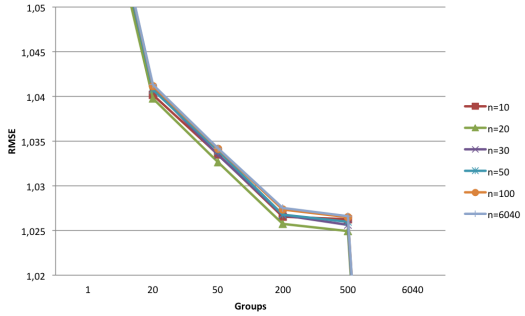

Fig. 6. Detail to study parameter *n*

2) Setting the Parameters of MergeRecommendations: Here, the experiments conducted to set the two parameters used by the system (i.e., $neighbors$ and n) are presented.

Selection of the number of *neighbors*. In order to predict a rating for a user, the users most similar to the one considered are selected. In order to do so, the right number of neighbors has to be selected 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. Fig. 7 and the underlying table, show the RMSE values for increasing values of $neighbors$. As highlighted in [18], 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, great improvements can be noticed. As expected, RMSE takes the form of a convex function (Fig. 8 shows a particular of Fig. 7), that indicates that after a certain value improvement stops. In these experiments, that value is 100.

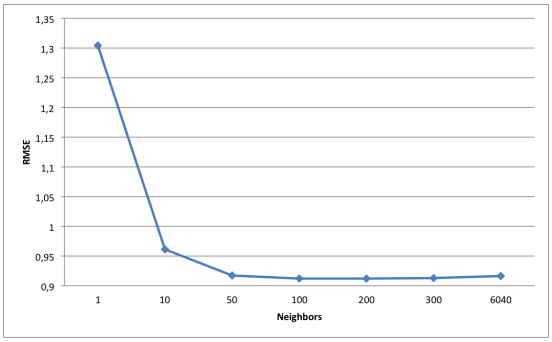

Fig. 7. RMSE for increasing number of *neighbors*

TABLE III. 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

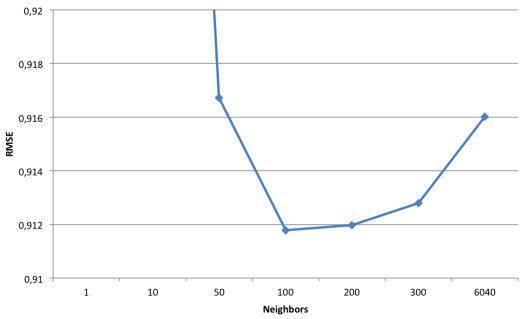

Fig. 8. RMSE takes the form of a convex function.

Independent-samples t-tests, conducted 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 ($M = 1.304622$, $SD = 0.00$) and 100 neighbors ($M = 0.911785$, $SD = 0.00$); $t(7.59) = 450.02$, $p = 0.00$. There is a also a significant difference in the RMSE values for 10 neighbors ($M = 0.961122$, $SD = 0.00$) and 100 neighbors ($M = 0.911785$, $SD = 0.00$); $t(7.41) = 54.44$, $p = 0.00$. A significant difference is also present in the RMSE values for 50 neighbors ($M = 0.916725$, $SD = 0.00$) and 100 neighbors ($M = 0.911785$, $SD = 0.00$); $t(7.97) = 6.02$, $p = 0.00$. The RMSE values present a difference for 100 neighbors ($M = 0.911785$, $SD = 0.00$) and 200 neighbors ($M = 0.911968$, $SD = 0.00$); $t(7.99) = 0.24$, $p = 0.82$. There is a also a difference in the RMSE values for 100 neighbors ($M = 0.911785$, $SD = 0.00$) and 300 neighbors ($M = 0.912803$, $SD = 0.00$); $t(7.97) = 1.06$, $p = 0.33$. There is a significant difference in the RMSE values for 100 neighbors ($M = 0.911785$, $SD = 0.00$) and 6040 neighbors ($M = 0.916022$, $SD = 0.03$); $t(7.99) = 1.27$, $p = 0.24$. 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 it is faster to compute predictions considering 100 neighbors instead of 200, $neighbors = 100$ is the value chosen for the algorithm.

Choice of the top-n items. This set of experiments evaluates how big the list of recommendations made for each user (i.e., the top-n) has to be, by testing parameter n. Fig. 9 and the underlying table, show that the choice of the top-5 ratings brings to the best results.

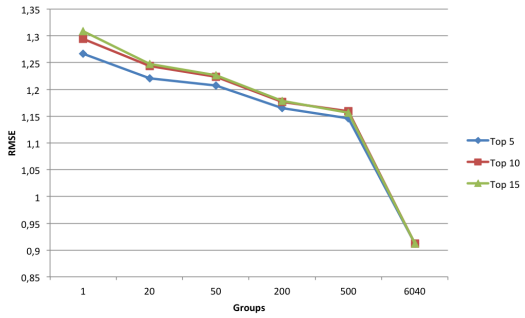

Fig. 9. RMSE values for increasing values of top *n* ratings

TABLE IV. 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 conducted, 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 $n = 5$ ($M = 1.266718$, $SD = 0.00$) and $n = 10$ ($M = 1.294696$, $SD = 0.00$); $t(7.74) = 3.39$, $p = 0.01$. For 20 groups, there is a difference between $n = 5$ ($M = 1.220748$, $SD = 0.00$) and $n = 10$ ($M = 1.243682$, $SD = 0.00$); $t(7.99) = 1.46$, $p = 0.18$. When 50 groups are considered, there is a difference between $n = 5$ ($M = 1.207548$, $SD = 0.00$) and $n = 10$ ($M = 1.223508$, $SD = 0.00$); $t(7.98) = 0.65$, $p = 0.53$. For 200 groups, there is also a difference between $n = 5$ ($M = 1.16532$, $SD = 0.00$) and $n = 10$ ($M = 1.176224$, $SD = 0.00$); $t(7.99) = 0.54$, $p = 0.60$. With 500 groups, there is a difference between $n = 5$ ($M = 1.146128$, $SD = 0.00$) and $n = 10$ ($M = 1.159176$, $SD = 0.00$); $t(7.45) = 0.65$, $p = 0.54$. Results show that when the number of groups increases, the significance of the difference between the values decreases. 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 system.

3) Setting the Parameters of PredictionsAggregation: Since the algorithm used by $PredictionsAggregation$ to predict individual ratings is the same used by $MergeRecommendations$ and it was already tested, no experiments to set the parameters have to be conducted. 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 the underlying table.

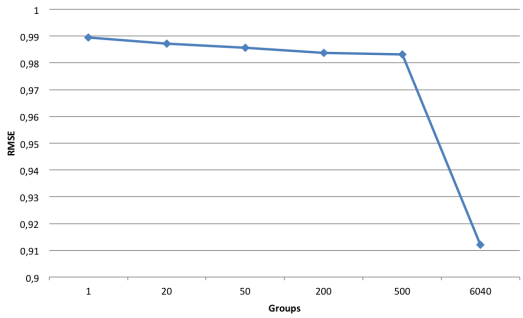

Fig. 10. RMSE values of *PredictionsAggregation*

TABLE V. 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) Selection of the best system: Fig. 11 and the underlying table, report the results obtained by each system with its best configuration. An aspect not previously deepened in the previous experiments, is that for all the systems, as the number of groups grows, the quality of the results improves. So as the number of groups increases, the RMSE values get lower. This means that the systems can have better performances when more recommendations can be produced.
The results obtained by three approaches show how the prediction tasks affects the quality of group recommendation. $MergeRecommendations$, the system that merges individual recommendations achieves the worst results. This is the sign that with automatically detected groups, if the preferences of a user 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 of

 ISBN: 978-1-61208-311-7

the figure. So, building predictions using a group model that uses an average to calculate predictions does not allow to capture the individual preferences and predict significant group ratings. At the bottom of the figure, with the best results, there is the system that merges individual preferences (i.e., $PredictionsAggregation$). 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 results. Independent-samples t-tests confirm that there is a significant difference between the RMSE values of $PredictionsAggregation$ and the ones obtained by the other systems. Results of the tests are not presented to facilitate the reading of the paper.

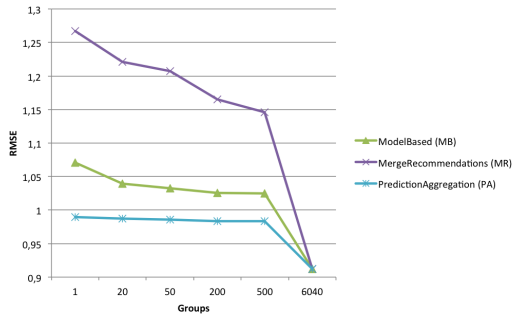

Fig. 11. RMSE obtained by the each system

TABLE VI. 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

V. CONCLUSIONS AND FUTURE WORK

This paper explored the ratings prediction task in group recommendation scenario in which groups are automatically detected. The experiments conducted allowed to achieve important contributions, now recapped.

- Exploring the ratings prediction task in this scenario allowed to understand that the use of all the predictions built for each user allows to achieve better results, with respect to the approaches that build predictions for a group or use the individual recommendations.
- Results improve as the number of groups grows. Analyzing the performances for different numbers of groups allowed to explore the trade-off between the number of recommendations built and the accuracy of the system.

Future work will focus on improving the cohesion of the groups. This will be done by adding more information to the input of the clustering algorithm, in order to detect groups with more homogeneous preferences and to produce more accurate group recommendations.

ACKNOWLEDGMENT

This work is partially funded by Regione Sardegna under project CGM (Coarse Grain Recommendation) through Pacchetto Integrato di Agevolazione (PIA) 2008 “Industria Artigianato e Servizi”.

REFERENCES

[1] F. Ricci, L. Rokach, and B. Shapira, “Introduction to recommender systems handbook,” in *Recommender Systems Handbook*. Berlin: Springer, 2011, pp. 1–35.

[2] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, 2005.

[3] A. Jameson and B. Smyth, “Recommendation to groups,” in *The Adaptive Web, Methods and Strategies of Web Personalization*. Springer, 2007, pp. 596–627.

[4] H. Lieberman, N. W. V. Dyke, and A. S. Vivacqua, “Let’s browse: A collaborative web browsing agent,” in *IUI*, 1999, pp. 65–68.

[5] Y. Zhiwen, Z. Xingshe, and Z. Daqing, “An adaptive in-vehicle multimedia recommender for group users,” in *Proceedings of the 61st Semiannual Vehicular Technology Conference*, vol. 5, 2005, pp. 2800–2804. [Online].

[6] Z. Yu, X. Zhou, Y. Hao, and J. Gu, “Tv program recommendation for multiple viewers based on user profile merging,” *User Modeling and User-Adapted Interaction*, vol. 16, no. 1, pp. 63–82, Mar. 2006. [Online].

[7] L. Ardissono, A. Goy, G. Petrone, M. Segnan, and P. Torasso, “Intrigue: Personalized recommendation of tourist attractions for desktop and hand held devices,” *Applied Artificial Intelligence*, vol. 17, no. 8-9, pp. 687–714, 2003.

[8] A. Jameson, “More than the sum of its members: challenges for group recommender systems,” in *Proceedings of the working conference on Advanced visual interfaces*. ACM Press, 2004, pp. 48–54.

[9] M. O’Connor, D. Cosley, J. A. Konstan, and J. Riedl, “Polylens: A recommender system for groups of user,” in *Proceedings of the Seventh European Conference on Computer Supported Cooperative Work*. Kluwer, 2001, pp. 199–218.

[10] S. Amer-Yahia, S. B. Roy, A. Chawla, G. Das, and C. Yu, “Group recommendation: Semantics and efficiency,” *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 754–765, 2009.

[11] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and M. A. Rueda-Morales, “Group recommending: A methodological approach based on bayesian networks,” in *Proceedings of the 23rd International Conference on Data Engineering Workshops, ICDE 2007*. IEEE Computer Society, 2007, pp. 835–844.

[12] J. B. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1. University of California Press, 1967, pp. 281–297.

[13] J. B. Schafer, D. Frankowski, J. L. Herlocker, and S. Sen, “Collaborative filtering recommender systems,” in *The Adaptive Web, Methods and Strategies of Web Personalization*. Springer, 2007, pp. 291–324.

[14] X. Amatriain, A. Jaimes, N. Oliver, and J. M. Pujol, “Data mining methods for recommender systems,” in *Recommender Systems Handbook*. Springer, 2011, pp. 39–71.

[15] J. Masthoff, “Group recommender systems: Combining individual models,” in *Recommender Systems Handbook*. Springer, 2011, pp. 677–702.

[16] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl, “An algorithmic framework for performing collaborative filtering,” in *Research and Development in Information Retrieval*. American Association of Computing Machinery, 8/1999 1999. [Online].

[17] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, “An efficient k-means clustering algorithm: Analysis and implementation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 881–892, July 2002. [Online].

[18] C. Desrosiers and G. Karypis, “A comprehensive survey of neighborhood-based recommendation methods,” in *Recommender Systems Handbook*. Berlin: Springer, 2011, pp. 107–144.

[19] Netflix Prize, “Frequently asked questions.” [Online]. Available: http://www.netflixprize.com/faq

 ISBN: 978-1-61208-311-7