Case-Based Aggregation of Preferences for Group Recommenders

Lara Quijano-Sánchez[1], Derek Bridge[2],
Belén Díaz-Agudo[1], and Juan A. Recio-García[1]
lara.quijano@fdi.ucm.es, d.bridge@cs.ucc.ie,
belend@sip.ucm.es, jareciog@fdi.ucm.es

[1] Department of Software Engineering and Artificial Intelligence,
Universidad Complutense de Madrid, Spain
[2] Department of Computer Science, University College Cork, Ireland

Abstract. We extend a group recommender system with a case base of previous group recommendation events. We show that this offers a new way of aggregating the predicted ratings of the group members. Using user-user similarity, we align individuals from the active group with individuals from the groups in the cases. Then, using item-item similarity, we transfer the preferences of the groups in the cases over to the group that is seeking a recommendation. The advantage of a case-based approach to preference aggregation is that it does not require us to commit to a model of social behaviour, expressed in a set of formulae, that may not be valid across all groups. Rather, the CBR system's aggregation of the predicted ratings will be a lazy and local generalization of the behaviours captured by the neighbouring cases in the case base.

1 Introduction

Groups often holiday together; tour museums and art galleries together; visit historic sights together; attend concerts and other events together; dine in restaurants together; watch movies and TV programmes together; listen to music together; cook and eat together. They must select the items which they intend to consume together, ranging from holiday destinations to recipes, in a way that reconciles the different preferences and personalities of the group members. For this, they may seek the support of a recommender system. But where the majority of recommender systems suggest items based on the preferences of an individual consumer, *group recommender systems* suggest items taking into account the preferences and personalities of the members of a group [4].

Commonly, group recommender systems aggregate predicted ratings for group members [4]: for each group member, a single-person recommender system predicts a set of ratings for the candidate items; then, the group recommender aggregates the ratings. The new group recommender system that we present in this paper takes the same approach, i.e. it aggregates the preferences of the group members, but it uses Case-Based Reasoning (CBR) for the aggregation. Figure 1 is suggestive of its operation. The system has a case base of past group

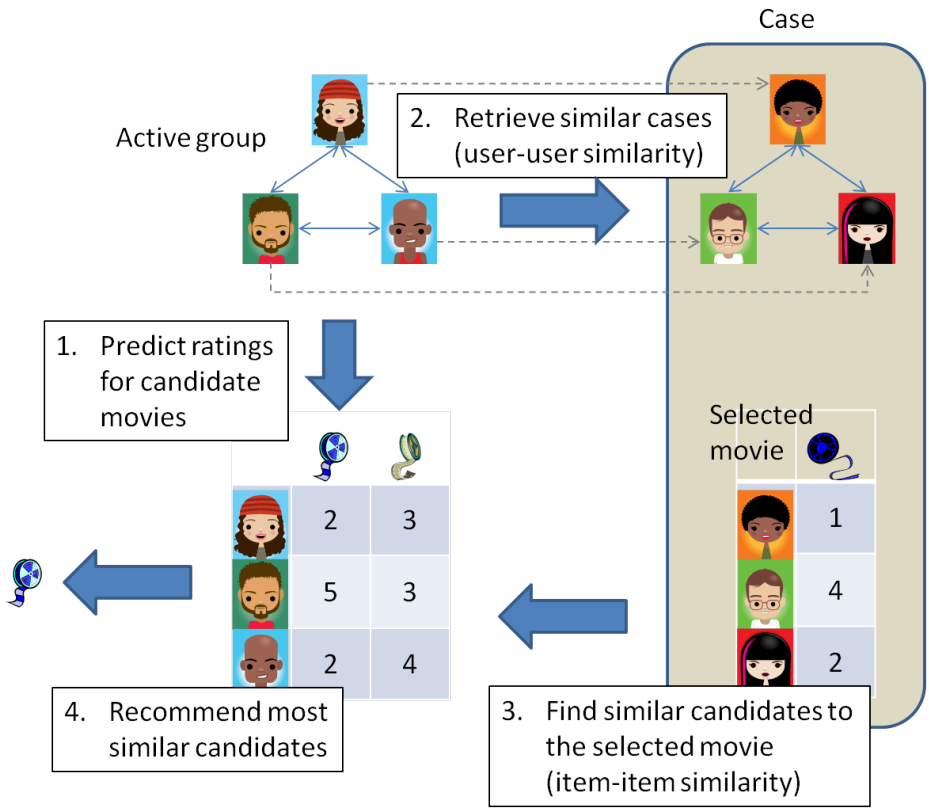

Fig. 1: Overview of the case-based recommender

recommendation events. Each case (right-hand side in the diagram) records the members of the group; the candidate items; the item that the group chose to consume together, which we will call the *selected item*; and the ratings that each group member gave to the selected item after consuming it. To make a recommendation to a new active group (top-left in the diagram), the CBR system deploys a unique combination of user-user and item-item similarity, as follows:

Step 1: First, it uses a user-based collaborative recommender to predict a rating for each candidate item by each group member.

Step 2: Next, it retrieves cases, i.e. past group recommendation events, that involve groups that are similar to the active group. Case retrieval uses the user-user similarity measure, and, as a by-product, it aligns each member of the active group with a member of the group in the case (the dashed lines in Figure 1). The similarity measure compares group members on their age, gender, personality and ratings and the degrees of trust between members of each group (the solid lines between group members in the diagram).

Step 3: Then, it reuses each case that is retrieved: the contributions that each group member made in choosing the selected item are transferred to the corresponding member of the active group. This is done by scoring the new candidate items by their item-item similarity to the selected item. In this way, the retrieved cases act as implicit models of group decision-making, which are transferred to the decision-making in the active group.

Step 4: Finally, it recommends the candidate items that have obtained the highest scores.

The paper explains this more fully. Section 2 gives some background exposition that we need for later sections; Section 3 describes an existing group recommender system, which we will use for comparison purposes; Section 4 describes the new case-based group recommender; Section 5 describes an experiment that compares the new recommender with the one we developed previously; and Section 6 concludes and presents some ideas for future work.

2 User-user and item-item similarity

Suppose there are n users, $U = \{u : 1 \dots n\}$, and m items (e.g. movies), $I = \{i : 1 \dots m\}$. Let r be a ratings matrix and $r_{u,i}$ be the rating that user u assigns to item i. Ratings are on a numeric scale, e.g. 1 = terrible and 5 = excellent, but $r_{u,i} = \perp$ signals that u has not yet rated i.

The similarity between one user and another, $u \in U, u' \in U, u \neq u'$, can be computed using Pearson Correlation [3], ρ. In effect, this computes the similarity between two *rows* in a ratings matrix like the one in the table in the lower left-hand part of Figure 1. The *user-user similarity* is:

$$\rho_{u,u'} \hat{=} \frac{\sum_{i \in I \wedge r_{u,i} \neq \perp \wedge r_{u',i} \neq \perp} (r_{u,i} - \bar{r}_u)(r_{u',i} - \bar{r}_{u'})}{\sqrt{\sum_{i \in I \wedge r_{u,i} \neq \perp \wedge r_{u',i} \neq \perp} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I \wedge r_{u,i} \neq \perp \wedge r_{u',i} \neq \perp} (r_{u',i} - \bar{r}_{u'})^2}} \quad (1)$$

$\bar{r}$ denotes a mean value and σ denotes a standard deviation, and these are computed over the *co-rated items* only ($i \in I \wedge r_{u,i} \neq \perp \wedge r_{u',i} \neq \perp$).

Suppose we want to recommend to active user u_a one or more of a set of candidate items $T_a \subseteq I$. For example, T_a could be the set of movies showing this week at u_a's local multiplex. Using user-user similarity, $\rho_{u,u'}$, we can build a *user-based collaborative recommender* [3, 13]. For each $i \in T_a$, it will predict active user u_a's rating for i, $\hat{r}_{u_a,i}$. It can do this using nearest-neighbour methods: from the users for whom $\rho_{u_a,u'}$ is greater than zero, it finds the k users $u' \in U$ who have rated i and who are most similar to u_a. The predicted rating is a weighted average of the neighbours' ratings for i [12]. The recommender suggests to the user the k' items $i \in T_a$ for which the predicted ratings $\hat{r}_{u_a,i}$ are highest.

But, given a ratings matrix we can equally well compute the similarity between one item and another, $i \in I, i' \in I, i \neq i'$, the *item-item similarity*, again using Pearson correlation. In effect, this computes the similarity between two

columns in a ratings matrix such as the one in the lower-left of Figure 1:

$$\rho_{i,i'} \;\hat{=}\; \frac{\sum_{u \in U \wedge r_{u,i} \neq \perp \wedge r_{u,i'} \neq \perp} (r_{u,i} - \bar{r}_i)(r_{u,i'} - \bar{r}_{i'})}{\sqrt{\sum_{u \in U \wedge r_{u,i} \neq \perp \wedge r_{u,i'} \neq \perp} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U \wedge r_{u,i} \neq \perp \wedge r_{u,i'} \neq \perp} (r_{u,i'} - \bar{r}_{i'})^2}} \tag{2}$$

In this case, the means ($\bar{r}$) and standard deviations (σ) are computed over the *users who have rated both items* ($u \in U \wedge r_{u,i} \neq \perp \wedge r_{u,i'} \neq \perp$).

Using item-item similarity, $\rho_{i,i'}$, it is possible to build an *item-based collaborative recommender* [6, 13], although we use it for a different purpose in this paper. Before presenting the case-based group recommender in detail, we present the group recommender system against whose performance we will be comparing the new recommender.

3 Social recommendations to groups

For the comparison, we use a group recommender that we developed previously [11, 8]. With real data and, in more recent work, with a larger dataset of artificial data, we showed that, relative to simpler approaches, our group recommender improves the accuracy of predicted group ratings and the precision of group recommendations, and that is why we use it here.

Let $G_a \subseteq U$ be an active group of users, in our case a group which intends to see a movie together. The goal is to recommend k' items from a set of T_a items. As Section 1 has mentioned, the system works by aggregation of ratings, as follows:

- For each $i \in T_a$ taken in turn, the recommender does the following:
  - It predicts a rating for item i, $\hat{r}_{u_a,i}$, for each individual group member $u_a \in G_a$. It does this using the user-based collaborative technique that we described in Section 2, i.e. it averages the ratings of i given by u_a's k most similar neighbours who have rated i.
  - It applies a function, designated dbr (which stands for *delegation-based rating*), to each predicted rating. The dbr function modifies $\hat{r}_{u_a,i}$ to take into account the *personality* of the user and the strength of connections between this person and other members of the group, which we refer to as their *trust*. In this way, not all the predicted individual ratings will contribute equally in the aggregation. We explain it in detail below.
  - It aggregates the individual predicted ratings into a single group rating $\hat{r}_{G_a,i}$. Possible aggregation functions include *least misery* (where the minimum is taken), and *most pleasure* (where the maximum is taken) [7]. We experimented with both before [9], and we found *most pleasure* to give better results, and so we adopt that here:

$$\hat{r}_{G_a,i} \;\hat{=}\; \max_{u \in G_a} \mathrm{dbr}(\hat{r}_{u,i}, G_a) \tag{3}$$

- It recommends the k' items in $i \in T_a$ for which the predicted group ratings $\hat{r}_{G_a,i}$ are highest.

The delegation-based method recognizes that a person's opinions may be based in part on the opinions of other members of the group. The formula, which we explain below, is as follows:

$$\text{dbr}(\hat{r}_{u,i}, G_a) \hat{=} \frac{\sum_{v \in G_a \wedge v \neq u} t_{u,v} \times (r_{v,i} + \theta_{r_{v,i}} \times (v.pers - u.per))}{\sum_{v \in G_a \wedge v \neq u} t_{u,v}} \quad (4)$$

In Equation 4, $t_{u,v}$ denotes the trust between u and v, which is a real number between 0.0 (no connection) and 1.0 (strong connection). In a real application, such as the Facebook movie group recommender that we have built [10], $t_{u,v}$ can be based on distance in the social network, the number of friends in common, relationship duration, and so on. As you can see, for user u in group G_a, we take into account the predicted ratings, $r_{v,i}$, for each other member of the group, $v \in G_a, v \neq u$, weighted by the trust between the two users, $t_{u,v}$. This follows [2], where a method for group recommendations using trust is proposed.

In Equation 4, *u.pers* denotes user u's personality, also a real number between 0.0 (very cooperative) and 1.0 (very selfish). In our Facebook group movie recommender, users complete a personality test on registration. The details of the test are in [15]. In Equation 4, the rating given by another group member $r_{v,i}$ is increased or decreased depending on the difference in personality, $v.pers - u.pers$. This way, users with stronger personalities will contribute more to the final score. A user v with a positive opinion of i, i.e. where $r_{v,i}$ is greater than the mid-point of the ratings scale, will want to increase u's opinion of i; but if v has a negative opinion, i.e. where $r_{v,i}$ is less than the mid-point of the scale, then v will want to decrease u's opinion. We model this through a function θ:

$$\theta_{r_{v,i}} \hat{=} \begin{cases} 5 & \text{if } r_{v,i} \geq mid \\ -5 & \text{otherwise} \end{cases}$$

where *mid* is the mid-point of the ratings scale, e.g. 3 on a five-point scale. We chose the constants (5 and -5) because the mean difference in personality values is 0.2 and therefore the impact of $\theta_{r_{v,i}}$ in Equation 4 will typically be 1 or -1.

4 A Case-Based Group Recommender System

Our new group recommender takes a case-based reasoning approach. There are two motivations for a case-based approach to group recommender systems.

- Firstly, groups tend to recur: the same group (with few variations) repeats activities together. Furthermore, group structures tend to recur: in the case of movies, for example, family outings comprising two adults and two children are common, as are parties of friends in the same age range.
- Secondly, group recommenders such as the one described in Section 3, have a 'one-size-fits-all' approach to the way they combine the predicted individual ratings. This ignores the possibility that different groups might have very different dynamics, not captured by a single theory expressed in a set of

formulae that apply globally. A case-based approach does not require us to commit to a model of social behaviour and to find a way to express that model in a set of formulae. Rather, aggregation of predicted ratings will be a lazy and local generalization (in the spirit of CBR) of the behaviours captured by the neighbouring cases in the case base.

4.1 Case representation

Assume a case base CB in which each case $c \in CB$ records a previous group recommendation event. Each case will have the following structure:

$$\langle id_c, \langle G_c, T_c \rangle, i_c, \{r_{u,i_c} : u \in G_c\} \rangle$$

- id_c is a case identification number, used to distinguish the case from others.
- The *problem description* part of the case comprises:
  - $G_c \subseteq U$, the group of users who used the recommender previously. For each user $u \in G_c$, we will know demographic information such as u's age ($u.age$) and gender ($u.gender$); u's ratings, $r_{u,i}$ for some set of items; and u's personality value, $u.pers$. And, for each pair of users $u \in G_c, v \in G_c, u \neq v$, we will know the trust value, $t_{u,v}$.
  - $T_c \subseteq I$, the set of items that the users were choosing between. In our cases, these were the movies that were at the local multiplex on the occasion when this group used the recommender.
- The *solution* part of the case contains just $i_c \in T_c$, the selected item, i.e. the item that the group agreed on. In our cases, this is the movie that the group went to see together.
- The *outcome* part of the case $[1, 5]$ is a set of ratings. These are the actual ratings r_{u,i_c} that the members of the group $u \in G_c$ gave to item i_c: for example, after a group has gone to see their selected movie, group members return and rate the movie. In practice, some members of the group will not do this. In these cases, we can use $\hat{r}_{u,i_c}$ instead, i.e. the rating that a user-based collaborative recommender (Section 2) predicts the user $u \in G_c$ will assign to i_c. However, we have not so far evaluated empirically the consequences of using predicted ratings in place of actual ratings.

We now explain how this recommender makes its recommendations.

Step 1: Predict individual ratings

As usual, the goal is to recommend k' items from a set of items, $T_a \subseteq I$, to an active group of users, $G_a \subseteq U$. We can write the problem statement as $PS = \langle G_a, T_a \rangle$. The first step is to predict individual ratings $\hat{r}_{u,i}$ for each candidate item $i \in T_a$ for each member of the active group $u \in G_a$. We do this using a standard user-based collaborative recommender, as described in Section 2.

Later in the process, it may be necessary to insert virtual users into G_a, i.e. ones that are not real people. We explain when and why this happens at the

appropriate time. But it simplifies the later exposition if we say now how we predict the ratings of items by virtual users. Since virtual users have no actual ratings, we cannot use the user-based collaborative recommender, as we do for real users. Instead, if u is a virtual user, its predicted rating for item i, $\hat{r}_{u,i}$, is the population average rating for i: $\frac{\sum_{u \in U \wedge r_{u,i} \neq \perp} r_{u,i}}{|u \in U \wedge r_{u,i} \neq \perp|}$.

Step 2: Retrieve cases

The next step is to find the k'' *most similar cases*. We use $k'' = 3$. The similarity between a problem statement $PS = \langle G_a, T_a \rangle$ and a case $c = \langle id_c, \langle G_c, T_c \rangle, i_c, \{r_{u,i_c} : u \in G_c\} \rangle \in CB$, $\text{sim}(PS, c)$, is calculated on the basis of group similarity:

$$\text{sim}(\langle G_a, T_a \rangle, \langle id_c, \langle G_c, T_c \rangle, i_c, \{r_{u,i_c} : u \in G_c\} \rangle) \hat{=} \text{gsim}_{cbr}(G_a, G_c) \quad (5)$$

This means that in our work case similarity only takes the groups, G_a and G_c, into account; it does not take into account the items, T_a and T_c. T_c contains the items that G_c contemplated in the past, but T_a contains items that G_a are contemplating right now, e.g. movies that have just come to town. These sets may or may not overlap. If they do, we have the basis for a refinement to the similarity we could use in case retrieval. We leave this to future work.

We denote the group similarity by gsim_{cbr}, and we emphasize that this is a new definition, richer than definitions that we have used in other work [8]. In effect, it is a form of graph similarity: users are nodes; trust relationships are weighted edges.

In our definition of group similarity, we pair each user from the active group G_a with exactly one user from the group in the case G_c and vice versa. In other words, we will be finding a *bijection* from G_a to G_c. This raises a problem when comparing groups of different sizes, where a bijection is not possible. In such situations, we could simply say that $\text{gsim}_{cbr}(G_a, G_c) = 0$. However, we did not want to do this. It might force the system to retrieve unsuitable cases. Consider a case base that just happens to contain many families of four (two adults, two children), no families of five, but many parties of five friends. If the active group is a family of five (two adults, three children), it is surely not appropriate to prevent retrieval of families of four and only retrieve parties of five friends.

To enable comparisons, this is the point, prior to computing similarity, that we insert additional virtual users into either G_a or G_c, whichever is the smaller, in order to make the groups the same size.

Now, we can define the group similarity measure. Consider any pair of equal-sized groups, G and G' and a bijection, f, from G to G'. The function f will map members of G to G', and so for any $u \in G$, we can compute the similarity, psim_{cbr}, to his/her partner $f(u) \in G'$. We will do this for each user and his/her partner, and take the average:

$$\text{gpsim}_{cbr}(G, G', f) \hat{=} \frac{\sum_{u \in G} \text{psim}_{cbr}(u, f(u))}{|G|} \quad (6)$$

But, we also have trust values for each pair of users in G, and we can compute the similarities between each of these and the trust values for the corresponding pair of users in G'. Again we take the average (dividing by the number of pairs):

$$\text{gtsim}_{cbr}(G, G', f) \hat{=} \frac{\sum_{u \in G, v \in G, u \neq v} \text{tsim}_{cbr}(t_{u,v}, t_{f(u),f(v)})}{|G|^2 - |G|} \tag{7}$$

We combine gpsim_{cbr} and gtsim_{cbr} in a weighted average to obtain the following definition of the similarity between any pair of equal-sized groups, G and G', given a bijection f from G to G':

$$\text{gsim}_{cbr}(G, G', f) \hat{=} \alpha \times \text{gpsim}_{cbr}(G, G', f) + (1 - \alpha) \times \text{gtsim}_{cbr}(G, G', f) \tag{8}$$

We currently use $\alpha = 0.5$.

This definition of gsim_{cbr} (Equation 8) uses gtsim_{cbr} (Equation 7), which uses tsim_{cbr}, the similarity between two trust values, which we have not yet defined. We use their range-normalized difference:

$$\text{tsim}_{cbr}(x, y) \hat{=} \text{rn\_diff}_t(x, y) \tag{9}$$

where

$$\text{rn\_diff}_{attr}(x, y) \hat{=} 1 - \frac{|x - y|}{range_{attr}} \tag{10}$$

There is a problem, however. If one or both of u or v (Equation 7) is a virtual user, we will not have a trust value; similarly, if one or both of $f(u)$ or $f(v)$ is virtual. In these situations, we impute an average trust value between that pair of users, which empirically we found to be 0.05.

Equally, the definition of gsim_{cbr} (Equation 8) uses gpsim_{cbr} (Equation 6), which uses psim_{cbr}, the similarity between a person u in one group G and a person v in another group G', which we have not yet defined. We make use of their ratings, age, gender and personality values. Specifically, we combine local similarities into a global similarity. The local similarities are as follows. For the users' ratings, we use the Pearson correlation (Equation 1) but normalized to $[0, 1]$, denoted here by $\rho_{[0,1]}$. For gender, we use an equality metric:

$$\text{eq}(x, y) \hat{=} \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

For ages and personalities, we use the range-normalized difference. Finally, the global similarity, psim_{cbr}, is simply an average of $\rho_{[0,1]}$, eq_{gender}, rn_diff_{age} and rn_diff_{pers}.

Again we have the problem of virtual users, who do not have ages, genders, personalities, or ratings. If either user is a virtual user, we simply take psim_{cbr} to be the mid-point of the similarity range. Empirically, this is 0.6. This means that there is neither an advantage nor a disadvantage to being matched with a virtual user and, since everyone must be paired with someone, this seems appropriate.

While this completes the definition of $\text{gsim}_{cbr}(G, G', f)$, it assumes that we give it a particular bijection, f, which pairs members of G with members of G'. But, for the similarity, we want to consider *every* such bijection and settle on the *best one*, the one that gives the best alignment between the group members (their ages, genders, personalities, ratings) and the trust values. We must compute $\text{gsim}_{cbr}(G, G', f)$ for each bijection.

Let $\mathcal{B}(A, B)$ denote all bijections between equal-sized sets A and B. For example, if A is $\{a, b, c\}$ and B is $\{x, y, z\}$, then one bijection is $\{a \mapsto x, b \mapsto y, c \mapsto z\}$, another is $\{a \mapsto y, b \mapsto x, c \mapsto z\}$, and so on. Our definition of the similarity of group G and G' is based on finding the bijection, out of all the possible bijections, that maximizes $\text{gsim}_{cbr}(G, G', f)$:

$$\text{gsim}_{cbr}(G, G') \hat{=} \max_{f \in \mathcal{B}(G,G')} \text{gsim}_{cbr}(G, G', f) \tag{12}$$

Think of this as finding the pairing that maximizes total similarity. It does mean that a person in G might not be paired with the person who is most similar in G': it optimizes total similarity (over all group members and all trust values).

If G (and G') are of size n, then there are $n!$ bijections, and all must be considered. There is cause to be concerned whenever a computation requires consideration of $n!$ objects, because of the way that factorial grows with n. But, fortunately, the groups that most recommenders will deal with will be small enough to keep this manageable. For example, of 525 movie-going events reported to us through a Facebook poll, 21 were of size seven or a little above seven. Those that were of size seven would require consideration of $7! = 5040$ bijections, which remains manageable. If there are group recommenders where the number of bijections becomes too large, then some sort of sampling or greedy heuristic can be used, with the cost that the optimal bijection might be missed.

Step 3: Reuse cases

At this point, we have explained our similarity measure, which is used to retrieve the k'' most similar cases. We must now explain how we reuse the cases that we have retrieved. To simplify the explanation, we will first consider the reuse of a single retrieved case, denoted $c = \langle id_c, \langle G_c, T_c \rangle, i_c, \{r_{u,i_c} : u \in G_c\} \rangle$.

Immediately, there is an issue that we must resolve. We want to predict G_a's ratings for each $i \in T_a$. But in case c, the selected item (e.g. the movie which the members of G_c went to see), was chosen from among T_c, which in most cases will not be equal to T_a: group G_a is going to the movies this week, whereas group G_c describes a previous outing to the movies, when it is probable that a different set of movies were on show. How can we transfer the contributions that the members of G_c made to the selection of $i_c \in T_c$ to the new situation where members of G_a must select an item from T_a?

The key to this is item-item similarity, which we described in Section 2. With item-item similarity, we can find the item $i \in T_a$ that is, for these users, most similar to $i_c \in T_c$. But there remains a problem. The Pearson correlation between two items i and i' is computed over the users who have rated both i

	Predicted ratings for candidate movies	
G_a	**Shrek**	**Hulk**
Ann	2	3
Ben	5	3
Col	2	4

	Actual ratings for the selected movie
G_c	**Twilight**
Dee	1
Edd	4
Flo	2

(a) No users in common

	Predicted & actual ratings		
Aligned users	**Shrek**	**Hulk**	**Twilight**
Ann+Dee	2	3	1
Ben+Edd	5	3	4
Col+Flo	2	4	2

(b) Using the bijection

Fig. 2: How item-item similarity is used

and i' (Equation 2). There is no guarantee that there will be any user in either G_a or G_c who has rated both $i \in T_a$ and $i_c \in T_c$. But this is where the bijection f found in Equation 12 can be used again. When comparing a rating from a user $u \in G_a$ for an item $i \in T_a$, we can use the rating $r_{f(u),i_c}$ made by the corresponding user $f(u) \in G_c$ for the item $i_c \in T_c$. It is by this means that we transfer the contributions that users in c made in their group decision to the group decision for $\langle G_a, T_a \rangle$.

But there is still a problem. The users $u \in G_a$ are unlikely to have a rating $r_{u,i}$ for the items $i \in T_a$, because T_a contains the candidate items that the group is choosing between. Instead, we use their predicted ratings $\hat{r}_{u,i}$, which we computed previously (Section 4.1) or, in the case of virtual users, the population average rating for the item.

Figure 2 contains an example. Suppose Ann, Ben and Col are in active group G_a, and that Dee, Edd and Flo are in case G_c. Figure 2a shows that we are unable to compute the item-item similarity between the selected movie from the case, Twilight, with the candidate movies, Shrek and Hulk. The movies have no users in common. For the active group, we have the predicted ratings for the candidate items; for the group in the case, we have the actual ratings for the selected movie. But suppose that, by the bijection, Ann maps to Dee, Ben maps to Edd and Col maps to Flo. Then, we can compute the item-item similarity between Shrek and Twilight by comparing Ann's predicted rating for Shrek with Dee's actual rating for Twilight, and Ben's predicted rating for Shrek with Edd's actual rating for Twilight, and so on. In effect, while there may be no users in these two groups who have rated both Shrek and Twilight, we are treating Ann & Dee as a 'single person' who has a rating for both Shrek (Ann's predicted rating) and Twilight (Dee's actual rating); see the Figure 2b.

We use Equation 2 to do this, but there are some changes. First, instead of computing the correlation over all users U, we compute it only over the users

$u \in G_a$. Secondly, wherever the formula uses $r_{u,i}$, we now use u's predicted rating, $\hat{r}_{u,i}$; and wherever the formula uses $r_{u,i'}$, we now use the rating given by the user in G_c who corresponds to u, i.e. $r_{f(u),i'}$.

We must still decide what to do if the groups are not of the same size. Consider the situation first where G_a is smaller than G_c . When we computed group similarity gsim_{cbr} earlier, we will have inserted extra virtual users into G_a. In this situation, we would not use G_a in place of U in Equation 2; rather, we would use the augmented version of G_a in place of U. That way, we can properly transfer the decision of the larger group to the smaller group: each person's contribution in the larger group is transferred to someone, either a real person from the smaller group or a virtual person who was inserted into the smaller group.

In the situation where G_a is larger than G_c, we will have earlier inserted virtual users into $G_c$in order to compute gsim_{cbr}. This time, however, we do use G_a in place of U. In other words, we compute the item-item similarity only on the ratings of the real people in G_a and their real counterparts in G_c. The virtual users were obviously not in reality present when G_c made its decision to consume i_c, so it makes no sense to transfer their contributions (i.e. none) to the decision-making of the smaller group G_a. This is achieved by simply computing item-item similarity over the real users and their counterparts, which is what Equation 2 will do if we use G_a in place of U. This does mean that, in these situations, there will be users in G_a whose opinions will be ignored (because they have no real counterparts in the smaller group, G_c).

So, we have explained how, given a retrieved case c, we can compute the similarity between i_c from c and each $i \in T_a$. We repeat this for each of the k'' retrieved cases. We can accumulate the item-item similarities and weight them by the group similarities. Formally, if C is the set of k'' cases, then the score for a candidate item $i \in T_a$ is $\sum_{c \in C} \text{gsim}_{cbr}(G_a, G_c) \times \rho_{i,i_c}$.

Step 4: Recommend items

All the items in T_a have now received a score based on cumulating the similarities to the selected items in similar cases, weighted by the degree of similarity to those cases. So, finally, we recommend the k' items that have the highest scores.

5 Experiment

5.1 Group Recommender Dataset

We need a dataset with which we can evaluate our new system. We have built a social group recommender as a Facebook application [10]. But, at the time of writing, it cannot provide the volume of data that we need for conducting experiments. Unfortunately, neither are we aware of a public dataset for group recommenders. Hence, we created our own dataset. We have explained its construction elsewhere [8], and so we only summarize here.

We created our dataset from the MovieLens 1M dataset (`www.grouplens.org`), which gives us around 1 million ratings on a scale of 1 to 5 for around 6040 users for nearly 4000 movies. We created 100 groups from the MovieLens users, selecting group members at random but in such a way that everyone in a group falls into the same age range, and we ensured that there were at least 15 movies which are co-rated by all members of the group. When we create cases, these 15 movies will be the set T_c. We created 50 groups of size 2, 18 of size 3, 16 of size 4, 7 of size 5, 5 of size 6, and 4 where we took the size to be 7, this distribution being based on respondents to a Facebook poll that we administered.

The MovieLens dataset gives us the age, gender and ratings of each user. We had to impute personality values, which we did using the population distributions given in [15, 14]. Similarly, we had to impute trust values between pairs of users in the same group. We took the trust between users u and u' to be the number of movies on whose ratings they agree as a proportion of the movies that either of them has rated. We take it that users agree if both have given the movie a rating above the ratings mid-point (which is 3) or if both have given the movie a rating below the ratings mid-point.

As we have explained, we have engineered matters so that, for each group, there is a set of 15 movies that all members of the group have rated, and we are treating these 15 movies as T_c, the set of movies that this group was choosing between. (Remember that T_c can be different for every group.) To create a case, we need to indicate which of these 15 movies the group will actually have chosen to go to see. For this, we got the opinion of four 'experts', two for each group. The experts voted on which three movies in T_c the group was most likely to select, placing movies into first, second and third position. Depending on the level of agreement between the experts, there might be ties for, e.g., first place, and so, although there were only three positions, the sets contained between three and five movies. We will designate this ordered set by E (for 'Expert') and we will use E_1 to mean movies in the first position in E, E_2 to mean movies in the first and second positions in E, and so on.

5.2 Evaluation Methodology

The dataset that we have created has 100 movie-going events, in other words 100 cases. We use a leave-one-out cross-validation methodology, where we remove each case in turn from the case base and present it to the recommenders.

We use three recommenders in these experiments: *Std*, *Soc* and *CBR*. *Std* is a simple group recommender: it uses the user-based collaborative recommender to predict the ratings each group member would give to the candidate items, and combines the ratings using the principle of *most pleasure*. *Soc* does the same but, before aggregation, it uses extra social data to modify individuals' predictions using the *delegation-based* method of Section 3. *CBR* is the new recommender, which uses cases to aggregate predicted ratings, which we described in Section 4.

Recall that each recommender recommends the top $k' = 3$ movies from the 15 candidates. Let R be the set of recommendations made by a particular recommender. Then we want to compare R with E from above. We computed total

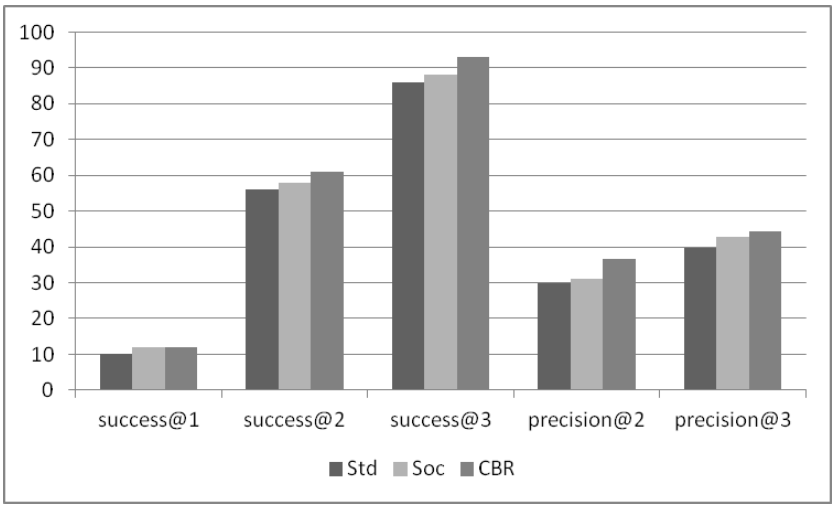

Fig. 3: Results of the experiment

success@n for $n = 1, 2, 3$, where *success*@$n = 1$ if $\exists i, i \in R \wedge i \in E_n$ and is 0 otherwise. For example, when using *success*@2, we score 1 each time there is at least one recommended movie in the top two positions of E. We also computed total *precision*@n for $n = 1, 2, 3$, where *precision*@$n \hat{=} |\{i : i \in R \wedge i \in E_n\}|/n$. For example, if no recommended movie is in the top two positions in E, then *precision*@2 = 0; if one recommended movie is in the top two positions in E, then *precision*@2 = 0.5.

5.3 Results

Figure 3 shows *success*@n for $n = 1, 2, 3$ and *precision*@n for $n = 2, 3$ (*precision*@1 = *success*@1 and is therefore not shown).

The first observation about the results is that, as n gets bigger, the results get better, e.g. *success*@2 results are better than *success*@1 results. This is not surprising: with bigger n, it is simply easier to make a recommendation that matches an expert judgement. The second observation is that results with *Soc* are better than results with *Std*: the use of the social information improves the quality of the recommendations. This is not a new result [11, 8]. But what is new, our third observation, is the performance of the CBR system. It is never worse, and usually better, than both of the other systems. In detail, *CBR* has the same total *success*@1 (and *precision*@1) as *Soc*, just 12: it is very difficult for the systems to recommend the movie(s) the experts place in first position. But in all other cases, the CBR does better. For example, *Soc*'s *success*@2 = 58 but *CBR*'s *success*@2 = 61; and *Soc*'s *precision*@2 = 31 but *CBR*'s is 36.5. This shows the value of abandoning *Soc*'s single model of social behaviour, in favour of the lazy and local generalization that we obtain from the Case-Based

Reasoning. We suspect that the differences would be even more marked in real datasets with more variability in the make-up of the groups.

6 Conclusions

We have described a new case-based group recommender system. It aggregates the predicted ratings of members of the active group but with reference to ratings of users in similar cases. A user-user similarity measure aligns members of the active group with members of the group in the case. The system uses an item-item similarity measure to transfer the contributions made to the group decision from the case to the corresponding users in the active group. One of its advantages is that preferences will be aggregated in different ways depending on how they played out in neighbouring groups, rather than according to a global, hypothesized theory of social interaction. This is borne out by our experiment, in which the CBR system is never worse, and is usually better, than a system that has a global model of group behaviour, expressed as a set of equations.

In our experiment, the selected item(s) in the cases are chosen by experts with knowledge of the actual ratings. So they are, in some sense, the absolutely best item(s). Therefore, it makes sense to run an experiment in which we see the extent to which the systems recommend such items.

But, matters are more complicated in practice. Suppose the recommender has recommended a movie to a group, and the group members have come back and rated that movie. We cannot simply retain this as a case in the case base. It may be suboptimal; it may not have been the best movie for this group. If we retain it, we will replay it in any future recommendation where it gets retrieved as a neighbour, where it may contribute to suboptimal decisions in the future.

In fact, this is not just a problem with CBR in group recommenders. It is a more general problem for the evaluation of group recommenders. It is very difficult to know whether they make good recommendations or not. If a user watches a recommended movie in a group and later gives it a low rating, this does not mean that the group recommender has done a poor job. It may even be that the group recommender predicted that this user would give a low rating. But the movie was recommended nonetheless, as it was judged to be the one that best reconciled the different tastes and personalities of the group members.

The implication is that, when group recommenders seek feedback from group members after recommended items have been consumed, they may need to solicit two types of feedback: the opinion of each individual user about whether the item satisfied him/her or not, but also the opinion of each individual user about whether the item satisfied the group as a whole or not.

Even if we get this more nuanced kind of feedback, it is not clear at this stage how to use it in evaluation of recommenders or in building case-based recommenders, not least because different group members may disagree on whether the recommendation satisfied the group or not. In case-based recommenders, the *outcome* part of the case might need to become much richer, to capture the opinions of the group members after they have consumed the item together,

implying additional complexity in the kind of case-based recommender that we have described. This is a major issue for future work.

Other future work includes the use of datasets in which we explicitly arrange for the same group (or nearly the same group) to consume items together on a frequent basis, which can lead to a case base with more directly relevant cases in it. We hope too to gather more data from our Facebook group recommender and use this in future experiments.

References

1. D. Bridge. The virtue of reward: Performance, reinforcement and discovery in case-based reasoning. In H. Muñoz-Avila and F. Ricci, editors, *Procs. of the 6th International Conference on Case-Based Reasoning*, LNAI 3620, page 1, 2005.
2. J. Golbeck. Generating predictive movie recommendations from trust in social networks. In *iTrust: 4th International Conference on Trust Management*, pages 93–104, 2006.
3. J. L. Herlocker. *Understanding and Improving Automated Collaborative Filtering Systems*. PhD thesis, University of Minnesota, 2000.
4. A. Jameson and B. Smyth. Recommendation to groups. In *The Adaptive Web, Methods and Strategies of Web Personalization*, Lecture Notes in Computer Science, 4321, pages 596–627. Springer, 2007.
5. J. L. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 1993.
6. G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.
7. J. Masthoff. Group modeling: Selecting a sequence of television items to suit a group of viewers. *User Modeling and User-Adapted Interaction*, 14(1):37–85, 2004.
8. L. Quijano-Sánchez, D. Bridge, B. Díaz-Agudo, and J. A. Recio-García. A case-based solution to the cold-start problem in group recommenders. In *this volume*, submitted.
9. L. Quijano-Sánchez, J. A. Recio-García, and B. Díaz-Agudo. An architecture for developing group recommender systems enhanced by social elements. In *International Journal of Human-Computer Studies*. in press, 2012.
10. L. Quijano-Sánchez, J. A. Recio-García, B. Díaz-Agudo, and G. Jiménez-Díaz. Happy movie: A group recommender application in facebook. In *24th International Florida Artificial Intelligence Research Society Conference, FLAIRS- 2011*, 2011.
11. L. Quijano-Sánchez, J. A. Recio-García, B. Díaz-Agudo, and G. Jiménez-Díaz. Social factors in group recommender systems. In *ACM-TIST, TIST-2011-01-0013*. in press, 2011.
12. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Procs. of the Conference on Computer Supported Collaborative Work*, pages 175–186, 1994.
13. J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web, Methods and Strategies of Web Personalization*, Lecture Notes in Computer Science 4321, pages 291–324. Springer, 2007.
14. N. A. Schaubhut. *Technical Brief for the THOMAS-KILMANN CONFLICT MODE INSTRUMENT*. CPP Research Department, 2007.
15. K. W. Thomas and R. H. Kilmann. *Thomas-Kilmann Conflict Mode Instrument*. Tuxedo, N.Y., 1974.