A Case-Based Solution to the Cold-Start Problem in 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 potential solution to the cold-start problem. Suppose a group recommendation is sought but one of the group members is a new user who has few item ratings. We can copy ratings into this user's profile from the profile of the most similar user in the most similar group from the case base. In other words, we copy ratings from a user who played a similar role in some previous group event. We show that copying in this way, i.e. conditioned on groups, is superior to copying nothing and also superior to copying ratings from the most similar user known to the system.

1 Introduction

Restaurants; tourist attractions; vacation destinations; movies, music & TV when broadcast in shared spaces. All these are examples of items that can benefit from *group recommender systems*, i.e. recommender systems whose suggestions take into account the preferences of the members of a group of people who will consume the items together [4]. Group recommenders typically work by either (a) merging the recommendations that would be made to the group members, (b) aggregating the predicted ratings of the group members, or (c) constructing a group preference model from the preferences of the group members [4].

In this paper, in the context of movie recommendation to groups of friends, we consider a group recommender system that takes the second of these approaches. It runs, and aggregates the results of, a *single-person recommender system* for each member of the group. Specifically, it runs a *user-based collaborative recommender system* [3] to predict movie ratings for each member of the group. It finds a neighbourhood of users who have similar movie ratings to those of the active user; it predicts user ratings for candidate movies that neighbours have rated but which the active user has not rated. The group recommender aggregates the predicted ratings for each group member to arrive at ratings and thence suggestions that it can make to the group as a whole. Methods for aggregating ratings are reviewed in [5] and it is the *most pleasure* principle (see Section 3) that we use.

It is well-known that collaborative recommenders suffer from *cold-start problems* [3, 13]. In particular, a user-based collaborative recommender finds it difficult to make good predictions for new users for whom it has few ratings: it cannot reliably find neighbours who have similar ratings to those of the new user. The group recommender inherits this problem too because it aggregates the predicted ratings for each group member. Solutions to the cold-start problem for single-person recommenders are summarized in [13]. Solutions include: non-personalized recommendations for cold-start users using population averages; intelligent ways to solicit more ratings (e.g. [2, 11]); and hybrid recommenders that resort to content-based recommendations when there are insufficient ratings to make collaborative recommendations (e.g. [1, 6]).

The contribution of this paper is to introduce and evaluate a *case-based reasoning* (CBR) solution to this problem. We use a case base in which each case records a previous group movie recommendation event. When a group requests a new recommendation but where one or more of the group members is in cold-start, we find a case that describes a previous recommendation event where there are users who are not in cold-start but who play similar roles in their group to the roles the cold-start users play in the active group. We copy ratings from the users in the case to the corresponding users in the active group and only then proceed to run the single-person recommender and to aggregate its results. It is natural to use a CBR approach because, in the movie domain and similar domains, similar events recur: the same group (perhaps with some small variations) repeats activities together; and some age, gender and personality distributions will tend to recur too (e.g. two adults with two children, or several friends in the same age range).

Case-based reasoning (CBR) has been used in recommender systems before (e.g. [12]) and explicit parallels between CBR and user-based collaborative recommenders have been drawn (e.g. [7]). But we are unaware of any previous use of CBR in group recommenders or in solutions to the cold-start problem.

Section 2 defines a single-person user-based collaborative recommender system; Section 3 describes two group recommenders that aggregate the predictions made for each group member by the single-person recommender; Section 4 describes how we have extended these group recommenders to use a case base of previous group recommendation events to solve the cold-start problem; Section 5 proposes systems against which the case-based system can be compared; Section 6 describes the dataset that we have used in our experiments; Section 7 presents our experimental method; Section 8 contains results; and Section 9 concludes and presents some ideas for future work.

2 Single-person user-based collaborative recommenders

As we have explained, our group recommender runs a user-based collaborative recommender for each person that is a member of the active group. Although the operation of user-based collaborative recommenders is well-known, we summarize it here in order to be explicit and to introduce some notation.

Suppose there are n users, $U = \{u : 1 \dots n\}$, and m items (e.g. movies), $I = \{i : 1 \dots m\}$. Let $r_{u,i}$ 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.

Suppose we want to recommend to active user u_a one or more of a set of candidate target items $T_a \subseteq I$. For example, T_a could be the set of movies showing this week at u_a's local multiplex. The user-based collaborative recommender that we use works as follows [3, 13]:

- For each $i \in T_a$,
  - The similarity between the active user u_a and each other user $u \neq u_a$ who has rated i, is computed using Pearson Correlation [3], ρ.
  - After computing the similarity between u_a and each other user u who has rated i, the k nearest neighbours are selected, i.e. the k for whom $\rho_{u_a,u}$ is highest. In our work, we use $k = 20$ and we only include neighbours for whom $\rho_{u_a,u} > 0$.
  - A predicted rating $\hat{r}_{u_a,i}$ for active user u_a and target item i is computed from the neighbours' ratings of i as follows:

$$\hat{r}_{u_a,i} \hat{=} \bar{r}_{u_a} + \frac{\sum_{u=1}^{k}(r_{u,i} - \bar{r}_u)\rho_{u_a,u}}{\sum_{u=1}^{k}\rho_{u_a,u}} \tag{1}$$

- Having computed $\hat{r}_{u_a,i}$ for each $i \in T_a$, the system recommends to the active user the k' items from T_a whose predicted ratings are highest. We use $k' = 3$.

3 Group recommenders

Let $G_a \subseteq U$ be an active group of users, in our case a group who intend going to see a movie together. The goal again is to recommend k' items from a set of T_a items. We will do this by computing a predicted rating $\hat{r}_{G_a,i}$ for active group G_a and each target item $i \in T_a$ and then recommending the k' items in T_a that have the highest predicted ratings.

3.1 Standard group recommenders

As we have explained, a common approach to group recommendation, and the one that we follow, is to aggregate the predicted ratings of the members of the group, $\hat{r}_{u_a,i}$ for each $u_a \in G_a$ for the various i in T_a. Possible aggregation functions include *least misery* (where the minimum is taken) and *most pleasure* (where the maximum is taken). We experimented with both before [8], and we found *most pleasure* to give better results, and so we adopt that here:

$$\hat{r}_{G_a,i} \hat{=} \max_{u_a \in G_a} \hat{r}_{u_a,i} \tag{2}$$

We compute $\hat{r}_{G_a,i}$ for each $i \in T_a$ and recommend the k' with the highest aggregated predicted rating. We will designate this recommender by *Std*.

3.2 Social group recommenders

Our previous work showed an improvement in the accuracy of predicted group ratings by taking into account the *personality* of the users in the group and the strength of their connections, which we refer to as their *trust* [10]. We refer to our recommender that takes this extra social information into account as being *social* and the method it uses as being *delegation-based*

We obtain the personality of each user u, denoted $u.pers$, by making group members complete a personality test on registration with the recommender. The details of the personality test are in [15]. In a real application, such as the Facebook social group recommender that we have built [9], trust between users u and v ($u \in U, v \in U, u \neq v$), $t_{u,v}$, can be based on distance in the social network, the number of friends in common, relationship duration, and so on.

Using the *most pleasure* principle again, we have:

$$\hat{r}_{G_a,i} \hat{=} \max_{u_a \in G_a} \text{dbr}(\hat{r}_{u_a,i}, G_a) \tag{3}$$

Here the *most pleasure* principle is not applied directly to individual predicted ratings, $\hat{r}_{u_a,i}$. The ratings are modified by the dbr function, which takes into account personality and trust values within the group G_a to compute what we call a delegation-based rating (dbr).

Space limitations preclude a detailed description of the operation of dbr but it is described in [10]. In essence, it is a weighted average of multiple copies of $\hat{r}_{u_a,i}$, one copy for each other member of $u \neq u_a$ in group G_a. The weights are based on the trust between u_a and u, $t_{u_a,u}$, and a value that is computed from the difference in their personalities, $u_a.pers - u.pers$.

The recommender recommends the k' items i from T_a for which $\hat{r}_{G_a,i}$ is highest. We will designate this recommender by *Soc*.

4 Using CBR in recommenders for users in cold-start

As we have explained, an active user with few ratings is said to be in cold-start. The problem that this causes for the kind of recommenders that we have been discussing is that it becomes difficult to find a reliable neighbourhood of similar users from which predictions can be made. One solution is to copy some ratings into the profile of the active cold-start user from a similar user who has additional ratings. Similarity in this case (i.e. for finding a user from whom ratings can be copied) would be measured using demographic information (age, gender, etc.) [13] because the active user has insufficient ratings to find a similar user using Pearson correlation, ρ. Let v be the user who is similar to u_a and from whom ratings will be copied. Then u_a obtains ratings for all items i that v has rated ($r_{v,i} \neq \perp$) but that u_a has not ($r_{u_a,i} = \perp$).

A group recommender can take the same approach when members of the group are in cold-start: prior to predicting individual ratings, it can augment the ratings profiles of group members who are in cold-start with ratings that are copied from the profiles of similar users. But in a group recommender, we can

go further than using just demographic information for finding the most similar users from whom ratings will be copied. In our work, we investigate how to reuse ratings from similar users in similar groups in a case-based fashion.

4.1 Case representation

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

$$\langle id_c, \langle G_c, T_c \rangle, i_c \rangle$$

- id_c is a case identification number, used to distinguish the case from others, but otherwise not used by the CBR.
- 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 case, 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 item that the group agreed on. In our case, this is the movie that the group went to see together.

Cases could also contain some of the numbers calculated when making the recommendation to the group, for example, the predicted individual ratings, $\hat{r}_{u,i}$ for each $u \in G_c$ and for each $i \in T_c$. Or, cases could also contain the *actual* ratings that users assign to item i_c. In other words, having gone to see movie i_c, users may come back to the system and give an actual rating, r_{u,i_c}. We leave the possible exploitation of this additional information to future work.

4.2 CBR for cold-start users in groups

We will summarize the process by which the case base is used for cold-start users. Details of the similarity measures will be given in subsequent sections. As usual, the goal is to recommend k' items from a set of items, $T_a \subseteq I$, to an active group of users, $G_a \subseteq U$. The recommender will recommend the k' for which the predicted group rating, which is aggregated from the predicted individual ratings, is highest. Of course, if none of the users in G_a is in cold-start, then the system will work either in the fashion described in Section 3.1 or in the fashion described in Section 3.2.

But suppose, on the other hand, that one or more members of G_a are in cold-start. We define this simply using a threshold, θ: a user u_a is in cold-start if and only if the number of items s/he has rated is less than θ, $|\{i : r(u_a, i) \neq \perp\}| < \theta$. In this case, we need to use the CBR. For each user who is in cold-start, we will copy ratings from the *most similar user in the most similar group* in the case base. The details follow.

Case retrieval We can write the problem statement as $PS = \langle G_a, T_a \rangle$. We will find the *most similar case*, c^*, in the case base:

$$c^* \hat{=} \arg\max_{c \in CB} \text{sim}(PS, c) \tag{4}$$

The similarity between a problem statement $PS = \langle G_a, T_a \rangle$ and a case $c = \langle id_c, \langle G_c, T_c \rangle, i_c \rangle \in CB$, $\text{sim}(PS, c)$, is calculated on the basis of group similarity:

$$\text{sim}(\langle G_a, T_a \rangle, \langle id_c, \langle G_c, T_c \rangle, i_c \rangle) \hat{=} \text{gsim}(G_a, G_c) \tag{5}$$

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 is 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.

Case reuse Next, for each user u_a in G_a who is in cold-start, we find the *most similar user* u^* in case c^* who has rated movies that u_a has not. Let G^* be the group of people described in case c^*. We find:

$$u^* \hat{=} \arg\max_{u \in G^* \wedge \exists i, r_{u_a,i} = \perp \wedge r_{u,i} \neq \perp} \text{psim}_{CB}(u_a, G_a, u, G^*) \tag{6}$$

In the case of more than one such user, we choose the one from whom we can copy the most ratings, i.e. the one who has most ratings for movies that u_a has not rated. Then, temporarily (for the purposes of making u_a's prediction for the items in T_a), we copy into u_a's profile the rating for each item i that u^* has rated ($r_{u^*,i} \neq \perp$) that u_a has not ($r_{u,i} = \perp$).

With each cold-start user's profile augmented in this way, we can then proceed to compute group recommendations in the fashion described in Section 3.1, which we will designate by *Std-CB*, or in the fashion described in Section 3.2, which we will designate by *Soc-CB*. But, it should now be less problematic finding neighbourhoods for the users who are in cold-start because they now have augmented user profiles.

4.3 The most similar group

As we saw above, case retrieval in this system finds the most similar case to the problem statement, which is the one that contains the group that is most similar to G_a. This requires a definition of group similarity, gsim. We compute the similarity of any pair of groups, G and G', from the similarity of the users in the two groups, $\text{psim}_{CB}(u, G, u', G'), u \in G, u' \in G'$. We will define $\text{psim}_{CB}(u, G, u', G')$ in the next subsection.

So, the similarity of G to G' is the average similarity of each user u in G to his/her most similar user in G':

$$\text{gsim}(G, G') \hat{=} \frac{\sum_{u \in G} \text{psim}_{CB}(u, G, u^*, G')}{|G|} \tag{7}$$

where

$$u^* \hat{=} \arg\max_{u' \in G'} \text{psim}_{CB}(u, G, u', G') \tag{8}$$

Note that the mapping from users $u \in G$ to users $u' \in G'$ is not bijective, meaning we do not prevent two or more people from G being associated with the same user $u' \in G'$. This fact allows us to easily compare groups of different sizes without further complications. It does mean that, if two or more users from G_a are in cold-start, they may all copy ratings from the same user $u' \in G$. (We could have taken the option of requiring bijective mappings, either by only comparing equal-sized groups or by introducing 'virtual' users to make groups equal-sized, and we have done this in on-going work. But it seemed an unnecessary and costly complication in our work on cold-start.)

4.4 The most similar user

Our CBR solution to the cold-start problem in group recommenders requires a definition of the similarity between two users, u and u', in different groups, $\text{psim}_{CB}(u, G, u', G')$ where $u \in G$ and $u' \in G'$. This plays two roles in the CBR. First, as Section 4.3 explains, it is used in *case retrieval*, since *the most similar user* is part of the definition of *the most similar group*. Second, as Section 4.2 explains, it is used in *case reuse*, since ratings are copied to each cold-start user from his/her corresponding *most similar user* in the most similar case.

To define $\text{psim}_{CB}(u, G, u', G')$, the similarity between two users in groups, we make use of their ratings, their demographic information (age and gender) and the social information (personality and trust). Specifically, we compute local similarities for each of these, and then combine them into a global similarity.

The local similarities are as follows. For their ratings, we use the Pearson correlation but normalized to $[0, 1]$, denoted here by $\rho_{[0,1]}$. For gender, we use an equality metric and for ages and personalities, we use the range-normalized difference:

$$\text{eq}(x, y) \hat{=} \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} \qquad \text{rn\_diff}_{attr}(x, y) \hat{=} 1 - \frac{|x - y|}{range_{attr}} \tag{9}$$

For trust values, we compute the average trust value between user u and all other members of his group, $v \in G, u \neq v$, which we will denote by $\bar{t_u}$. Similarly, we compute the average trust value for the other user, $\bar{t_{u'}}$, and we use rn_diff to give the similarity of these two values. We do the same for the standard deviations of the trust values, σ_{t_u} and $\sigma_{t_{u'}}$. The global similarity, psim_{CB}, is simply an average of $\rho_{[0,1]}$, eq_{gender}, rn_diff_{age}, rn_diff_{pers}, $\text{rn\_diff}_{\bar{t}}$ and $\text{rn\_diff}_{\sigma_t}$.

5 Other recommenders for users in cold-start

An obvious question is whether it makes a difference that our case-based solution to the cold-start problem in group recommenders works on a group basis at all.

Why copy ratings from the most similar user in the most similar group? Why not copy ratings simply from the most similar user in the case base as a whole? Or why not copy ratings from the most similar user known to the system? Systems that work in these different ways will be useful for comparisons in our experiments, hence we define both of these more precisely now.

Consider the set of users who appear in at least one case in the case base:

$$U_{CB} \hat{=} \{u : \exists c = \langle id_c, \langle G_c, T_c \rangle, i_c \rangle \in CB \wedge u \in G_c\} \tag{10}$$

When trying to predict group G_a's rating for an item $i \in T_a$, then for any user $u \in G_a$ who is in cold-start, we could find, and copy ratings from, the most similar user in U_{CB}:

$$u^* \hat{=} \underset{u \in U_{CB} \wedge \exists i, r_{u_a,i} = \perp \wedge r_{u,i} \neq \perp}{\arg\max} \text{psim}_{U_{CB}}(u_a, u) \tag{11}$$

This is different from first finding the most similar case (in other words, the most similar group) and then, for each active user in cold-start, copying ratings from the most similar user in that group. Our case-based approach is conditioned on the groups; this alternative is not. Note that this alternative needs a new definition of the similarity between two people, $\text{psim}_{U_{CB}}$ in place of psim_{CB}. Above, we were able to compute and compare the average and standard deviations of the trust values between a user and all other members of his/her group. In this new setting, this no longer makes sense, since we are ignoring the groups. Hence, the global similarity $\text{psim}_{U_{CB}}$ will be the average of just $\rho_{[0,1]}$, eq_{gender}, rn_diff_{age} and rn_diff_{pers}. We will designate this recommender by *Std-UCB* (where it works in the fashion described in Section 3.1) and by *Soc-UCB* (where it works in the fashion described in Section 3.2).

The second of our two alternative cold-start recommenders ignores the case base altogether. It simply finds, and copies ratings from, the most similar user in U (the entire set of users), wholly ignoring whether they have previously participated in group recommendations or not. Hence,

$$u^* \hat{=} \underset{u \in U \wedge \exists i, r_{u_a,i} = \perp \wedge r_{u,i} \neq \perp}{\arg\max} \text{psim}_{U}(u_a, u) \tag{12}$$

Note that for the experiments in this paper, this requires yet another definition of the similarity between users, psim_U. This is because we only have personality values for users who have participated in group recommendation events. Hence, the global similarity psim_U will be the average of just $\rho_{[0,1]}$, eq_{gender} and rn_diff_{age}. We will designate this recommender by *Std-U* (where it works as per Section 3.1) and by *Soc-U* (where it works as per Section 3.2).

6 Group Recommender Dataset

We need a dataset with which we can evaluate our case-based solution to the cold-start problem in group recommenders. We have built a social group recommender as a Facebook application [9]. But, at the time of writing, it cannot

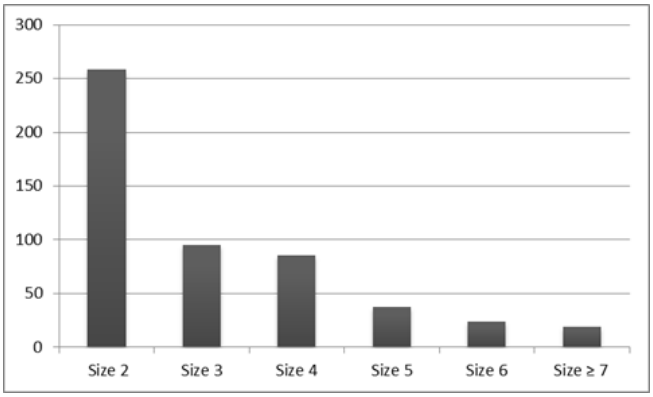

Fig. 1. Group sizes for 525 real movie-going events

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, and we explain how we did this here.

Base dataset We have used the MovieLens 1M dataset (`www.grouplens.org`). It gives us around 1 million ratings on a scale of 1 to 5 for around 6040 users for nearly 4000 movies. Each user has at least 20 ratings. The dataset also gives a small amount of demographic information about each user. In particular, we use the user's gender and age range (under 18, $18-24$, $25-34$, and so on).

Groups We created 100 groups from the MovieLens dataset. Group members are chosen at random from all users in the MovieLens dataset but subject to the following restrictions:

- In a group, users are distinct (but a user may be in more than one group).
- In a group, we ensure that all the users are in the same age range.
- In a group, we ensure that there are 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. These ratings themselves are withheld from the recommender, because it would not in general know a user's actual ratings for the movies that the group was choosing from.

We conducted a Facebook poll in which we asked respondents to tell us, for the last five times that they went to the cinema in a group, how large the group was. There were 105 respondents and so we learned the group size for 525 events (although we we cannot be certain that they were all distinct events). Figure 1 shows the distribution. We used the frequencies from this distribution to create our 100 groups. Hence, we have 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.

Personality values We had to impute personality values to the users in the groups. The personality test that we have described in previous work is the Thomas-Killmann Conflict Mode Instrument (TKI) [15]. Questions on the test reveal the extent to which a person uses each of five modes for dealing with conflict, including "competing", "compromising", "avoiding" and so on. These five modes can be summarized to give scores on two dimensions, "assertiveness" and "cooperativeness", from which we define a single numeric value, $u.pers$, in the range $[0, 1]$, where 0 signals a very cooperative person and 1 signals a very selfish person [10].

To impute personalities to users in our dataset, we make use of the population norms that the TKI Technical Brief provides [14]. We randomly give to each user five scores, one for each mode, based on the distributions given in the Brief. We calculate $u.pers$ from these.

We recognize that this is imperfect. Although the distribution of the five modes among our users will reflect the distribution in the population, the distribution within groups may not reflect reality. Because of the randomness, we might end up with a group of, for example, four very selfish people, where perhaps this rarely occurs in reality. We should be able to take a more informed approach in the future, once our Facebook application generates more data.

Trust values As we have discussed, in our Facebook application, trust is computed from Facebook data (distance in the social network, etc.), but that is not available to us for the users in the MovieLens dataset. Rather than simply imputing trust values at random, we have chosen to base them on ratings. For these experiments, the trust between users u and u' is the number of movies on whose ratings they agree as a proportion of the movies that either of them has rated. Agreement here is defined quite loosely: they 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. The formula is as follows:

$$t_{u,u'} \hat{=} \frac{|\{i : (r(u,i) > 3 \wedge r(u',i) > 3) \vee (r(u,i) < 3 \wedge r(u',i) < 3)\}|}{|\{i : r(u,i) \neq \perp \vee r(u',i) \neq \perp\}|} \quad (13)$$

Hence, in our dataset, trust is based on the degree of shared taste.

This does not mean that, when psim_{CB} combines $\rho_{[0,1]}$ with $\text{rn\_diff}_{\bar{t}}$ and $\text{rn\_diff}_{\sigma_t}$, it is counting the same shared ratings twice. $\rho_{[0,1]}$ compares ratings between members of different groups (*inter-group*); it aligns a person in one group with someone in the other group who has the same tastes. But $\text{rn\_diff}_{\bar{t}}$ and $\text{rn\_diff}_{\sigma_t}$ compare ratings within groups (*intra-group*) to give trust values, which are then compared between groups; they align a person in one group with someone who has similar trust relationships in the other group.

The chosen movie So far, we have described how we have created 100 groups. 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 (although we

withhold the ratings from the recommender), 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. But we cannot ask random groups of MovieLens users to work out which of their 15 candidate movies they would have gone to see together.

We used four human 'experts' who were given all the information about a group's members G_c and the candidate movies T_c (including the actual ratings by the members of G_c for the items in T_c) and were asked to decide which of the movies the group would be most likely to settle on. Each expert evaluated 50 cases, hence each of the 100 groups was evaluated by two experts (not always the same two experts). Experts were asked to give an ordered list of the three movies from T_c that they thought the members of G_c would agree on.

Since each case is being decided by two experts, we needed a voting scheme to reconcile their judgements. A movie that an expert placed in first position was given three votes; a movie placed in second position was given two votes; and a movie placed in third position was given one vote. By adding up and ranking movies by their votes, we obtain a final ordered list of the movies that G_c would be most likely to see. For example, if both experts placed a movie $i \in T_c$ in first place, then it would receive six votes and would come first in the final combined ordering. But if one expert placed i in first position and $j \neq i$ in second position, but the other expert placed them in the opposite order, then both get five votes. The final ordered set will contain a minimum of three movies (where the experts agreed on the same set of three movies from T_c) and a maximum of six movies (where the two experts disagreed entirely). In fact, the latter never happened; final ordered sets are roughly evenly-split between those of size three and those of size four, plus a handful of size five. 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.

7 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 compare their recommendations with the judgements of the experts.

We use eight recommenders in these experiments: *Std*, *Soc*, *Std-CB*, *Soc-CB*, *Std-UCB*, *Soc-UCB*, *Std-U* and *Soc-U*. *Soc* (social) indicates that, before aggregation, the recommender uses extra social data to modify individuals' predictions using the *delegation-based* method of Section 3.2, whereas *Std* (standard) indicates that they do not as in Section 3.1. The second part of the name, if there is one, indicates how the recommenders handle cold-start users. The four options here are: they do nothing for cold-start users; they copy ratings from the most similar user in the most similar case (*-CB*, Section 4); they copy ratings from the

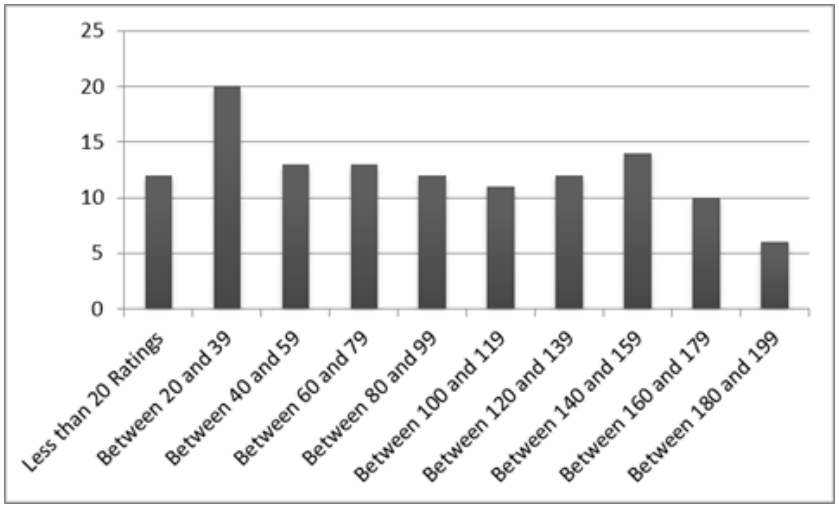

Fig. 2. Number of users in cold-start

most similar user from any case (-*UCB*, Section 5); or they copy ratings from the most similar user in the whole dataset (-*U*, also Section 5).

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, the ordered set of movies that the experts judged to be correct. We computed total *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.

We repeat the experiments with different cold-start thresholds. Figure 2 shows how many users are affected. We see that with $\theta = 20$, just over ten users are in cold-start; with $\theta = 40$, an additional twenty users are in cold-start; and then as θ goes up by 20, the number of users in cold-start grows by about an additional ten each time. (The threshold excludes the 15 ratings for T_a, which are withheld from the recommender.)

8 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) for cold-start threshold $\theta = 20$.

The first observation about the results is that, as one would expect, as n gets bigger, results improve but differences between systems become less pronounced:

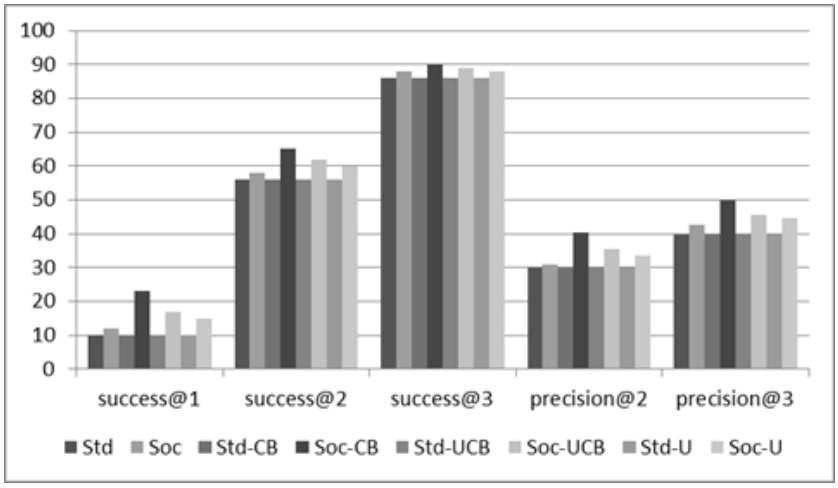

Fig. 3. Results for $\theta = 20$

with bigger n it is simply easier to make a recommendation that matches an expert judgement. The next observation comes from looking at pairs of bars. The first bar in each pair is a system that does not use social data, and the second is one that does. Consistently throughout all our results, systems that use social data out-perform their counterparts that do not, which shows the value of using personality and trust information. This is something we had already established in our previous work (e.g. [10, 8]), but it is good to see the result confirmed on our new dataset. A final (and the most important) observation is that the *Soc-CB* system out-performs the *Soc-UCB* system, which out-performs the *Soc-U* system, which out-performs the *Soc* system. In other words, a cold-start strategy that is conditioned on groups (from cases) copies ratings in a more informed and successful way than strategies that copy without regard to groups, and copying ratings is more successful than having no cold-start solution at all.

We tried out a similar cold-start solution in the context of a single-person recommender, where a single active user seeks movie recommendations. If the active user was in cold-start, we copied ratings from a similar user in U. Interestingly, doing so made no or almost no change to the *success*@n and *precision*@n results (not shown here) across several definitions of similarity. We conclude that, for our movie data, conditioning on groups really does seem to be the most effective way to use this cold-start solution.

Figure 4 shows the effects of varying θ from 20 to 200. In other words, more and more users are regarded as being in cold-start and are given ratings from other users. We only show systems that use social data because, as we have already said, they are better. The results for *Soc* itself remain the same for all values of θ because this system has no cold-start strategy. For the other systems, we see that results improve and then fall off as θ increases. For example, for *Soc-CB*, results improve until $\theta = 100$. For this system, 100 is the cut-off point: users

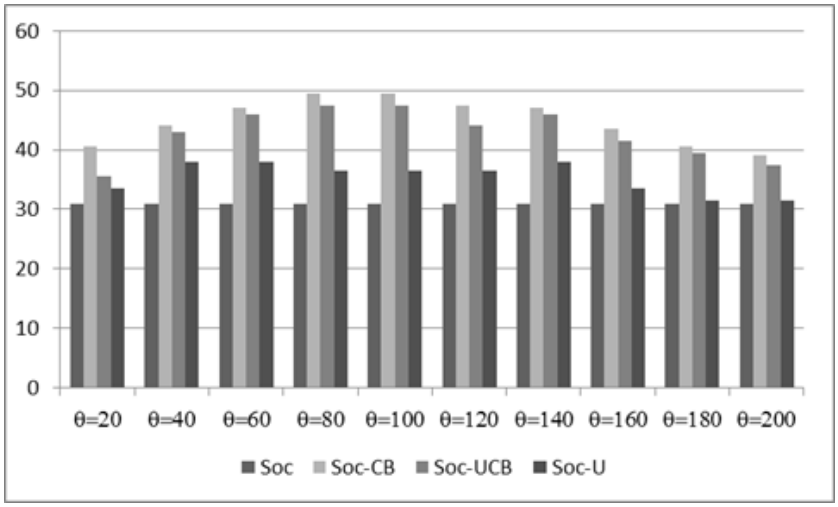

Fig. 4. Results for *precision*@2

with fewer than 100 ratings are ones we should regard as being in cold-start. A higher threshold treats so many users as being in cold-start that the tastes of the active group are swamped by the ratings copied from other users, causing system performance to decrease. The graph is for *precision*@2 but we observed the same pattern of results for all other measures.

9 Conclusions

We have presented a new solution to the cold-start problem in a collaborative group recommender. We use a case base of group recommendation events and copy ratings into the profile of users who are in cold-start from their most similar user in the most similar group in the case base. Our experiments on movie data show that, for users with fewer than 100 ratings, this strategy improves the quality of the group recommendations. The experiments also confirm, using new data, the results of our previous work, viz. a group recommender that uses social data, such as user personality and inter-personal trust, produces higher quality recommendations than one that does not use this data. A side-product of the research has been the construction of a dataset for group recommender research.

There is much that can be done to take this work forward. For us, the next step is to consider a case base in which we more explicitly arrange that there be cases (e.g. movie-going events) that involve groups whose members have a high degree of overlap with the members of the active group, so that we can experiment with the situation where the same group (or nearly the same group) consumes items together on a frequent basis. We also intend to consider richer case representations to take into account such things as timestamps, predicted

and actual ratings from group members, and the dynamics of reaching a consensus (e.g. changes in group membership and changes in the selected item). We hope too to gather more data from our Facebook application and use this data as the basis for future experiments.

References

1. M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. *Comms. of the Association of Computing Machinery*, 40(3):66–72, 1997.
2. N. Golbandi, Y. Koren, and R. Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *Fourth ACM International Conference on Web Search and Data Mining*, 2011.
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. 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.
6. P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Procs. of the Eighteenth National Conference on Artificial Intelligence (AAAI)*, pages 187–192, 2002.
7. D. O'Sullivan, D. C. Wilson, and B. Smyth. Improving case-based recommendation: A collaborative filtering approach. In S. Craw and A. Preece, editors, *Procs. of the 6th European Conference on Case-Based Reasoning*, pages 278–291. Springer, 2002.
8. 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.
9. 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.
10. 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.
11. A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the 2002 International Conference on Intelligent User Interfaces*, pages 127–134, 2002.
12. F. Ricci, B. Arslan, N. Mirzadeh, and A. Venturini. ITR: A case-based travel advisory system. In S. Craw and A. Preece, editors, *Procs. of the 6th European Conference on Case-Based Reasoning*, pages 613–641. Springer, 2002.
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.