T U M

I N S T I T U T F Ü R I N F O R M A T I K

Recommending for Groups in Decentralized Collaborative Filtering

Stefan Birnkammerer, Wolfgang Woerndl, Georg Groh

TUM-I0927
November 09

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-11-I0927-0/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

Contents

Abstract

This paper deals with the process of recommending movies for groups of people in a mobile, decentralized scenario. Many participating peers exchange their individual rating vectors and build up their own internal prediction model. A stationary device allows a group of peers to receive group recommendations and input explicit group ratings. After a brief introduction into the field of recommender systems, we show in particular the problems of group recommender systems and their evaluation. We propose two different approaches for generating movie recommendations to a group of people and introduce an extension to the usual datastructure of rating vectors in form of weighting factors in order to be able to take into account different types of rating information. An offline analysis of the presented algorithms by means of the MovieLens dataset is conducted afterwards and the results are presented and discussed. The paper concludes with a few recommendations on algorithm design based on our findings.

1 Introduction

Recommender Systems try to help their users to come to a decision in an extensive information domain. Typical examples of well known recommender systems are online shopping recommenders that suggest items such as CDs, books or movies, of which the system thinks that the user will like them. There has been a lot of research on the field of recommender systems in recent years regarding recommender quality, recommender speed, scalability, etc.

In most cases a recommender system is deployed on a centralized server, but if you care about the availability of the recommender system without a permanent connection to the server or if you are worried about privacy issues, then a decentralized, mobile recommender system might be worth to think about. Additionally, whereas in most recommender systems recommendations are generated for a single user (e.g. Online-Shopping), recommendations for not only one but a group of users are of special interest in decentralized recommender systems. This is, because many daily life activities are performed in groups of people, such as going to a restaurant or going to the movies, and most people always carry a mobile device with them, which are the predestinated candidates for the implementation of a decentralized recommender system.

In this paper we want to have a close look on the process of group recommendations in a decentralized mobile environment. The concrete scenario in this work is that of a decentral mobile recommender system for movies, i.e. a system which helps a (randomly gathered) group of people (where every member of the group owns a mobile device with the recommender system on it) to decide on movies to watch. Additionally, there should be a stationary device, where a group of users can input group ratings for movies to enhance recommender quality or receive group recommendations. See Figure 1 to get an idea how such a stationary device could look like.

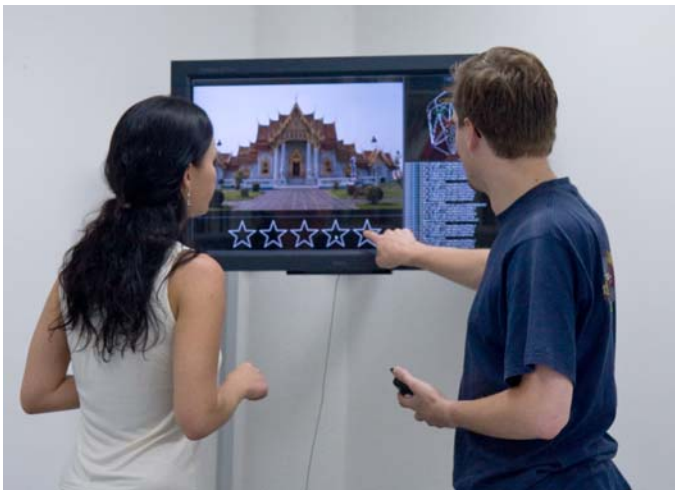


Figure 1: Stationary Device for input of group ratings (Source: [13])

The decision for a decentral, mobile recommender system puts several limitations on the way this recommender system should be implemented. Therefore, we pick up the work done in [13], which deals with the implementation of a decentral recommender system on mobile devices based on the previous work on PocketLens [14]. [13] uses an item-based nearest neighbor algorithm as described in section 2.2.2 to generate recommendations for the active user. This has the advantage that only an *item-item matrix* has to be managed by the mobile device and no rating vectors from other users have to be stored in the device's memory (see [14] for details how this can be done and [13] for further functional and storage improvements). If two mobile devices meet they share their rating vectors and recalculate their internal model on basis of this. Then a rating prediction for the active user (the mobile device's owner) can be computed by using the internal item-item matrix at every mobile device without the need of a connection to another device. Thus, also in our scenario, we limit ourselves to the use of an item-item matrix.

The key research question in this paper is how to deal with groups in the vicinity of the stationary device, respectively how to improve the group recommender quality. More practical we think of people who go to the movies or a DVD shop and stand in front of a device which tries to recommend movies to watch and also allows the group to rate movies.

Before proposing our solution we want to give a short introduction to conventional recommender systems. Afterwards we explain the challenges of group recommender systems in general and in particular with regard to our scenario. We then describe our solution and present the results of an evaluation of our solution by using the MovieLens dataset. Finally, we give a short summary and make a few proposals for future work.

2 Introduction to Recommender Systems

2.1 Basics

A recommender system helps the user to navigate through the items of a (very) broad domain, i.e. to select new items or to predict the satisfaction of the user with a selected item. Typical fields of application are for example product recommenders like the most prominent recommender system realized at Amazon.com [1] or movie recommenders like the academic research project MovieLens (http://www.movielens.org) [2]. All these recommender systems have in common that there are too many items in the application domain, so that the user cannot examine all of the items with regard to costs and time in order to finally come to a decision which item to choose. As already stated, there are two main tasks for recommender systems: presenting new interesting items to the user and predicting the user's satisfaction with a particular selected item. Obviously, picking new items is just a special case of predicting the user's rating for a single item. In this case predictions are generated for all available items and the n items with the highest predictions are presented to the user. So if we speak of the one or the other case henceforth, remember that both are eventually the same.

There has been a good amount of research in the field of recommender systems in recent years (for a more detailed overview as we will give in the following, which is beyond the scope of this paper, we recommend reading [3]). One of the key research questions in this area is about *how* the recommender systems generates its predictions about the user's satisfaction with a particular item. From now on we will speak of the *active user* when referring to the user, for whom recommendations are generated for. But before a computer system is able to actually compute any predictions it must be told somehow what items the user likes or dislikes. This is usually done by *user ratings*, which can be seperated again into *implicit* and *explicit* ratings (cp. [4]): With explicit ratings the user explicitely specifies his convenience with an item by selecting a rating on a scale (e.g. from 0 to 10 or from "hate it" to "love it"). On the other hand with implicit ratings the system tries to derive conclusions about the user by observing his behaviour. GroupLens for example, a recommender system for newsgroups, describes the use of the reading time of an newsgroup article as an implicit rating of this article by the user [5].

In any case, for every user u_i a so called *rating vector* r_i is defined, which specifies for every object n_j the rating $r_{i,j}$ of n_j by u_i. All rating vectors together finally build up to the *rating matrix* R. A great advantage of implicit ratings is, that they (at least partially) overcome the *sparsity problem*. Sparsity means the tendency that a user only rates a very small subset of the available objects in an application domain. Because the explicit rating of an item requires effort by the user and implicit ratings do not, implicit ratings result in a more dense rating matrix.

2.2 Computation of Predictions

With the rating matrix at hand, the next question is how to use this information to actually predict the user's satisfaction with items the user has *not rated yet*. There exist two main approaches for recommender systems:

2.2.1 Content-based Filtering

Probably the most basic approach of all is to look which items the active user likes and search for unrated items that are similar to them. If this similarity of items is defined by comparing objective item attributes, this approach is called *content-based filtering*. Content-based filtering has its roots in the field of *Information Retrieval* [6] and therefore is often used in cases where there is unstructured text associated with the objects. The computation of the similarity between objects can then be done for example by means of *term frequency* respectively *inverse document frequency*, which is about comparing documents by comparing the frequency of keywords within these texts, or *bayesian probability methods*, which reduce the problem of comparing documents to the problem of probabilistic classifying. Because content-based filtering to this extent is not used in this paper, we do not deal in detail with these methods but refer to [7] for an excellent overview of the topic.

2.2.2 Collaborative Filtering

Content-based filtering assumes that an item will be rated the same way as items that are similar to it. The second main approach in turn assumes that people who share the same preferences will also rate an item the same way. Due to the fact that this approach, contrary to content-based filtering, relies on the rating vectors of many people in order to compute the preference prediction for the active user, it is called *collaborative filtering*. Different classification paradigms exist to seperate between the different collaborative filtering approaches. We refer to the classification proposed in [8], that differentiates the approaches in *non-probabilistic* and *probabilistic* methods.

Arguably the most natural approach in non-probabilstic collaborative filtering with regard to the previously stated notion of collaborative filtering is that of finding a similarity neighborhood of users for the active user and take the ratings of this neighborhood for a particular item to finally compute a prediction value for the active user. This algorithm is called *user-based nearest neighbor* algorithm. To find the neighborhood of the active user n a similarity function $sim(n, u)$ that depicts the similarity of user n and u has to be defined. Then the neighborhood could for example be defined by taking every user u into account whose similarity $sim(n, u)$ with the active user n exceeds a certain threshold λ. Given the set of users in the neighborhood, denoted by $neighbors(n)$, the rating prediction of item i for the active user n could be computed as

$$pred(n, i) = \frac{\sum_{u \in neighbors(n)} \cdot r_{u,i}}{|neighbors(n)|}, \tag{1}$$

where $r_{u,i}$ denotes the rating of user u for item i. Further improvements would be the consideration of the different similarity values and the different rating behaviours of the users. We therefore take into account the similarity values and the average rating $\overline{r}_u$ of an user u:

$$pred(n,i) = \overline{r}_n + \frac{\sum_{u \in neighbors(n)} sim(n,u) \cdot (r_{u,i} - \overline{r}_u)}{\sum_{u \in neighbors(n)} sim(n,u)} \tag{2}$$

For a more detailed view on this equation and the following concerning collaborative filtering we recommend to have a look at [8]. Finally, the similarity function $sim(n,u)$ is often defined as the *Pearson Correlation*, first proposed in [9] ($CR_{n,u}$ denotes the items that both n and u has rated already):

$$sim(u,n) = \frac{\sum_{i \in CR_{n,u}} (r_{n,i} - \overline{r}_n)(r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i \in CR_{n,u}} (r_{n,i} - \overline{r}_n)^2} \sqrt{\sum_{i \in CR_{n,u}} (r_{u,i} - \overline{r}_u)^2}} \tag{3}$$

There exist several further improvements to the algorithm, which are covered and explained in [10].

A second nearest neighbor approach is the *item-based nearest neighbor* algorithm which relies on the notion that objects are similar to each other if their ratings by the useres are similar. Thus by defining a similarity function (for example a *pearson correlation* or a *cosine similarity*) an item-item similarity matrix can be generated. As already stated in the introduction this approach is well suited for the implementation in a decentralized recommender system, because the item-item matrix can be seen as an underlying abstract model, which can be used and maintained by every peer on its own to generate predictions. Therefore, [13] and also this paper follow the method proposed in [14] to build the item-item matrix which uses *vector cosine similarity*:

$$itemSim(u,w) = \cos \vec{u}, \vec{w} = \frac{\vec{u} \cdot \vec{w}}{||\vec{u}|| \cdot ||\vec{w}||} \tag{4}$$

Eventually the prediction for the active user is computed:

$$pred(n,i) = \frac{\sum_{j \in ratedItems(n)} itemSim(i,j) \cdot r_{n,j}}{\sum_{j \in ratedItems(n)} itemSim(i,j)} \tag{5}$$

Hereby $ratedItems(n)$ can stand for all items that n has rated. It is also possible and quite common to limit $ratedItems(n)$ to m items that n has rated with the highest similarity values $itemSim(i,j)$ for item i. In this case we speak of a *similarity neighborhood of n*. In the evaluation further below we limited the tested algorithms to a similarity neighborhood.

It is important to make clear the difference between content-based filtering and item-based filtering. The latter relies only on the ratings of the items and *not* on objective meta data like genre information, free text or anything else as it is the case with content-based filtering. Because item-based filtering first builds up an internal model (the item-item matrix) in order to later compute

predictions based on this model, these kinds of approaches are also often called *model-based collaborative filtering* in the literature.

As mentioned above, a further type of methods for collaborative filtering are probabilistic methods. A prominent example of this type are *bayesian networks*, as described in [10]. This paper does not deal with probabilistic algorithms and therefore does not cover this topic. Also the problems of collaborative filtering algorithms compared with content-based algorithms are not covered here, but later in the text when necessary.

3 Prelimenary Considerations on Group Recommender Systems

So far we gave a very short overview about the field of recommender systems. There has been plenty of work on this field since the mid 1990s regarding different algorithms for content-based and collaborative filtering [7, 8], techniques for evaluating the algorithms [2], actual evaluations and comparisons of different algorithms [10], the importance of explaining predictions to the user [11] and so on. But not until recently research in recommender systems was almost solely limited to generating predictions for *one* active user, although many items which are typical subjects to recommendation systems like music, movies or restaurants are enjoyed often by groups and not single individuals. One of the first systems that dealt with satisfying groups of people was MusicFX [12], which selects a music genre to be played in a gym on basis of the preferences of the currently training people. In the following we will cover the different issues that arise when generating recommendations for groups of people and review them with regard to the scenario of this paper.

Like in the field of usual recommender systems the classification of a group recommender system consists of many dimensions. Several different classification schemes were proposed by previous works, but eventually they all sum up to the dimensions we present in the following. For each dimension we analyze our scenario regarding this dimension.

3.1 Type of Group

The first question could be *what kind of group* the recommender system has to satisfy. [15] proposes classification in ephemeral/persistant and public/private. Of course a group of people watching a movie could be ephemeral, but in many cases you go to the movies with the same people for several times, so we assume persistant groups (but without a doubt the system has to support rapid ephemeral groups too). Although some applications of public groups could be imagined as useful (e.g. when allowing other groups to copy the own group's recommendations because the groups resemble each other), we assume private groups in our scenario due to privacy concerns.

3.2 Number of recommendations per group

A second important question is, whether the recommender system satisfies the group's demands only once or several times. Because in the previous paragraph we decided for persistant groups, the recommender in our scenario potentially has to deal with sequences of recommendations rather than single decisions. This is important because it raises the issue of order and fairness. According to [16], users tend to change the ratings of news items in a tv item sequence after having watched a first news clip (which influenced their mood). However, the consideration of the effects of the ordering of items in a sequence, especially the ordering of items like movies where one item is not necessarily watched instantly after the other, is too vague and more detailed research is beyond the scope of this work, so that we do not consider the ordering of the items here.

Fairness on the other hand is a more worthwhile topic. User studies as that in [16] and everyday experience show, that humans apply fairness strategies if they have to select a sequence of items for a group. You could imagine a case when the same group comes for the third time to the cinema and the recommender system explains: "The system recommends watching X because Elizabeth probably likes it a lot and the last times she was not that satisfied with the chosen movies as the other group members were."

[17] proposes three different methods for dealing with sequences of recommendations. The first possibility is to handle each recommendation as single problem, which makes it impossible to consider ordering or fairness with regard to the overall sequence. The second approach is the complete opposite by regarding the sequence of recommendations as one single problem. While this approach surely accounts for ordering and fairness it also implies that the system knows the final length of the sequence, which is not the case in our scenario. The third method is that one we also propose to use in our scenario: treating every recommendation on its own but taking into account the past recommendations. The FLYTRAP Music System [18] gives an example for the latter approach.

So one strategy in order to account for fairness could be to try to keep the average satisfaction of all group members approximately on the same level. If the system recognizes, that one group member has been discriminated so far, the welfare of this group member should be more important than the welfare of the others in the next recommendation.

3.3 Source of Information

A third classification dimension is that of the source of information. In our scenario a mobile device mainly collects information about the user's ratings by explicit ratings on a scale numeral scale. An additional source of information to the recommender system are the explicit ratings by the groups in front of the stationary device. Further below we will discuss how this source of information can be used for generating group recommendations and/or altering the user's rating vectors.

We could also think of yet another source of information, which could be used

for content-based filtering, if we assume that there is some sort of content information for every movie in the database. Because we will use the MovieLens dataset for evaluation of our thesis and this dataset provides genre information for every movie the utilization of this information could be quite useful in our scenario to overcome the problem of sparseness in collaborative filtering. A possible solution for example would be a hybrid approach that defines every rating $r_{n,i}$ of user n with $i \in unratedItems(n)$ as

$$r_{n,i} = \frac{\sum_{j \in genreItems(n,i)} r_{n,j}}{|\sum_{j \in genreItems(n,i)}|}, \tag{6}$$

i.e. as average of the ratings of items $genreItems(n,i)$ with the same genre as i that n has rated already. Of course these *genre ratings* should only be seen (and implemented) as computation aid and not as real user rating. By using this content information an user-based nearest neighbor algorithm could benefit from much more dense rating vectors when computing the user's neighborhood. This approach resembles *default voting*, a modification proposed in [10] that assigns a default voting value to items that either n or u have not rated when computing $sim(n,u)$. Empirical studies show that simple default voting is capable of improving recommender quality, so it would be plausible that the proposed hybrid approach using valuable information is also capable of doing so [10].

It would be an interesting question if this hybrid approach also enhances the quality of the item-item matrix in an item-based nearest neighbor algorithm, because [13] states that the density of the item-item matrix increases anyhow rapidly once a few normal rating vectors are integrated, so the potentially higher memory needs are probably of no consequence. With regard to real life experience our assumption is, that this indeed should enhance the quality because genre preferences *are* undoubtedly an important indicator for the probable rating of an unseen movie. We examine this question further below in the evaluation.

3.4 Generating Recommendations

3.4.1 How groups come to decisions

Before we discuss how a group recommendation should be generated in our scenario it is important to consider research about how groups come to decisions in general or respectively what is best for a group given the preferences of the group members. We will call this process the group's *decision strategy*. This question is studied in the field of *social choice theory*, which is strongly influenced by Arrow's impossibility theorem [20]. A commonly accepted set of desirable properties for voting systems exists (a group decision about ranking items can be seen as voting system with every group member "voting" with his preferences), but Arrow's theorem states that no voting system can have all of these properties at once. For a fuller treatment of some of these properties related to group recommendation see [16].

Social choice theory is the basis for the work conducted in [16] that is closely related to the discussion in our paper. [16] tries to evaluate by means of a user study which decision (voting) strategies real groups use to select a sequence of tv items when given the ratings of the items by every group member. Several strategies from social choice theory are proposed and eventually compared to the actual results of the user study. The key findings of this experiment are that real people seem to apply the *Average*, *Average Without Misery* and *Least Misery* strategies (though there is no clear dominant strategy) and care about fairness. We shortly introduce these three voting strategies:

- *Average:* the average of the individual ratings for every item is computed and used as the item's ranking
- *Average Without Misery:* same as Average, but dismisses items with at least one individual rating below a certain threshold
- *Least Misery:* rank the items by their minimum individual ranking

Least Misery takes into account that a group is always just as happy as the most unsatisfied group member. In our opinion this assumption is partially plausible but too strict in general. So for use in our scenario we opt for the approach of *Average Without Misery*. Imagine a group of ten people and two movies A and B of which A is rated with 9 (on a scale from 0 to 10) by all members but one who rates the movie with 4 and B is rated with 5 by all group members. *Least Misery* would rank movie B higher than movie A in contrast to *Average Without Misery*. We think that movie A should be preferred in this case, because though it is important to ensure that one group member is not dissatisfied completely, average group satisfaction is also very important.

This is especially true with regard to our scenario of a possible sequence of recommendations. As stated in [16] real people care about fairness and we will conduct this fairness consideration also in our scenario. So the user who is dissatisfied with the decision to watch a movie he rated with 4 and all others rated with 9 will have more weight in the recommendation process for the next item.

3.4.2 Existing approaches

Most research so far has focused on the question how group recommendations eventually should be computed. We follow the classification proposed in [17] that lists three different approaches:

- Merging of Recommendations Made for Individuals (RM)
- Aggregation of Ratings for Individuals (AR)
- Construction of a Group Preference Model / Aggregation of Profiles (AP)

In the following we will discuss each of the approaches with regard to our scenario.

Merging of Recommendations Made for Individuals This approach generates a set of recommendations for every group member (e.g. by using an usual individual recommendation algorithm) and then attempts to merge these sets in order to finally present a group set. The merging process could for example be a simple union operation. Like [17] we will not consider this approach because there is a broad consensus among researchers that this method is too simple and not a good choice for the generation of group recommendations.

Aggregation of Ratings for Individuals Another possible approach is to generate predictions for the unrated items of an user and use this information to calculate an aggregated group prediction for this item and respectively to rank the items. Eventually the system then computes a recommendation for the group as a whole.

This could be done by predicting the rating $r_{n,i}$ for every unrated item i of every group member $n \in G,\ G \subseteq N$ (e.g. by using one of the prediction algorithms described above) and finally compute the group's prediction for this item

$$R_i = aggregate(r_{n,i})\ n \in G$$

on basis of an aggregate function *aggregate*. GroupLens [5] for example uses the Least Misery strategy as aggregation function which results in $aggregate(r_{n,i}) = \min_j r_{n,i}$. Of course many aggregation functions are imaginable. Most strategies from social choice theory can be transformed to an aggregate function and especially those proposed in [16] (for a selection see the paragraph above).

Construction of a Group Preference Model The last approach in order to generate recommendations for groups is the attempt to find a profile that models the group's preferences as a whole, i.e. instead of using the individual user profiles to generate predictions the user profiles are used to generate a group preference model. This model can then be used to compute a group recommendation by means of an individual recommender algorithm.

An example for such a group preference model construction can be found in Let's Browse [21], one of the earliest group recommender research. The system combines the individual models, which consist of a keyword analysis of the user's homepages, by using a linear combination of those. The computed group model is henceforth used for making recommendations.

Also [13] uses this group preference model approach. The users' individual rating vectors are combined by a simple averaging operation over the items that at least one group member have rated to eventually form a group rating vector. This group rating vector is used subsequently to recommend items to the group based on the available item-item matrix.

3.5 Interaction with the system

[22] introduces another classification dimension: the ability of the system to actively support the final group decision or more generally, the ability of the system to let the group interact with it. A very interesting example is the Travel Decision Forum [23], which helps the users with finding a joint group preference model every group member agrees upon. For this purpose the system proposes values for the single dimensions of a preference specification form on basis of the individual data of the group members (every group member has to fill out the preference specification form at the beginning). Afterwards, all users have to agree upon a single way of filling out the form by interacting with the user interface, that displays the proposed value by the system and the preferences of the other group members. When this process is completed, the system can use a commonly agreed group preference model to compute group recommendations.

In our scenario we want to allow interaction with the system by the group in the following manner: a group of people arrives at the stationary device and likes to receive recommendations. The system hereupon makes a movie recommendation and presents it to the users. Now the group has the choice of either accepting the recommendation or dismissing it. In the case of dismissal the users should be allowed to rate the dismissed movie as a group. Another possible interaction with the system is a training mode which allows the users to rate as many random movies as they like as a group. Further below we discuss the effect of this scenario assumptions on the design of our approach.

3.6 Explaining Recommendations

As with usual individual recommender systems the explanation of recommendations to the group could be also of high value in the case of group recommender systems. We refer to [17] for a summary of different explanation approaches in different systems. We propose a way of explaining the recommendations generated by our approach on the spot below.

4 Our Approach

We now want to discuss our approaches for generating recommendations that are also used in the evaluation below. As to our knowledge, there are no significant comparative studies about the performance of the methods of aggregating ratings and aggregating profiles we will discuss two solutions for group recommendation in our scenario.

4.1 Constraints

Because of our concrete scenario, we limit the possible approaches by the constraint, that all approaches have to use an item-item matrix as proposed in [13]. Therefore the use of an item-based nearest neighbor algorithm is mandatory.

This has the advantage that every peer is able to manage and maintain its own recommendation model and does not have to store the rating vectors of other peers directly.

Although we had our concrete scenario with a group standing in front of the stationary device in mind, most of the computations discussed below can also be applied in order to make recommendations to a rapidly gathered, ephemeral group of users without the need of a stationary device. The adapatability heavily depends on the implementation, which is beyond the scope of this paper, but though we restrict our discussion to the scenario with a stationary device, in general every approach is applicable in an even more decentralized scenario as well.

We shortly summarize (and partially repeat) the pre-conditions both of the following approaches are based upon:

- There exists an item-item matrix A on every mobile device and on the stationary device
- The individual user rating vectors consist of explicitely stated preferences *and* the content-based genre information as discussed in 3.3

4.2 Aggregating Profiles

We begin our discussion with the creation of a group preference model, which is then used for the computation of group recommendations on basis of the item-item matrix A.

Aggregation Method The individual rating vectors must be combined somehow to form one group rating vector. For every item $i \in I_G$ in the subset $I_G \in I$ of items that at least one member of the group $G \subseteq N$ has rated already [13] sets the group rating

$$R_i = \frac{\sum_{n \in rated(i)} r_{n,i}}{\sum_{n \in rated(i)} 1}, \tag{7}$$

of item i as simple average of the individual ratings of the group members $rated(i)$ whose rating for item i is not null. Given the equality of all users (there is no user that is more important than another) this method is without a doubt plausible. However, one could argue, that this puts too much power towards the ratings of items that only a small subset, e.g. only one person, of the group members have rated. This very individual rating would have the same impact on the final group rating prediction as an averaged rating that was created from the ratings of all group members (see equation 8). It would be nice, if we could put more weight on those ratings that many of the group members have rated during the computation process, because the average of these ratings is potentially more significant to the estimation of the group's "real wish".

A similar problem occurs when thinking of the genre information that should be used in the user's rating vectors as discussed in 3.3. The just stated notion above also applies here, because genre information about films, yet probably

helpful, should be considered not as significant as the explicit ratings by the user.

In order to overcome this difficulty we propose an extension to the data structure that represents a rating vector. Until now the rating vector $\vec{r_n}$ of user n consists of a list of ratings for all items which the user has rated explicitely and those for which a genre rating is available. For every rating $r_{n,i}$ we now introduce a *rating weighting factor (rw factor)* $z_{n,i} \in \mathbb{N}$. The rw factors are eventually used in the computation of a rating prediction based upon the related rating vector:

$$pred(n,i) = \frac{\sum_{j \in ratedItems(n)} itemSim(i,j) \cdot r_{n,j} \cdot z_{n,j}}{\sum_{j \in ratedItems(n)} itemSim(i,j) \cdot z_{n,j}} \tag{8}$$

This modification of equation (5) and the basic data structure enables the algorithm to put emphasis on more significant rating data without the complete dismissal of less significant but valuable information. Please note that this affects not only the process of generating predictions but also the generation of the item-item matrix itself.

The function *itemSim* is "stored" in the underlying model, i.e. the item-item matrix. Due to the decentralized nature of our scenario every peer and the stationary device has its own internal model. Hence, the question arises which model should be used for the computation in equation 8. The item-item matrix of the stationary device could be considered more close to reality than the item-item matrices of the group members because technically it is a "power-user", who has contact to potentially much more people than a normal user. Another approach would be to extend the item-item matrix of everyone with a counter, which increases with every additional individual (respectively group) vector that is integrated into the model. Eventually the internal model with the highest counter, i.e. the highest number of integrated vectors, should be chosen as computation base.

Of course this is mostly a technical question regarding recommender quality and it is up to the implementation to deal with this consideration. For our discussion here it is of no practical relevance which item-item matrix is used after all.

Finally, plausible values for the rw factors must be found. For the individual rating vectors we denote

$$w_n = x \cdot \frac{|genreRatings(n)|}{|explicitRatings(n)|} \tag{9}$$

$$z_{n,i} = \begin{cases} w_n & if\ i \in explicitRatings(n) \\ 1 & if\ i \in genreRatings(n) \end{cases}. \tag{10}$$

By this definition the explicit ratings' impact on a computation of a prediction for the individual user is x times higher than the impact of the genre information. This is plausible because though the genre information is considered valuable the explicit ratings of the user should have the most impact on the

recommendation. The most beneficial value of x could be found by evaluation. Next, the merging algorithm that creates a group rating vector on basis of the individual group member rating vectors should compute a group rating vector that resembles the individual rating vector. In the first step the algorithm computes all group ratings R_i and the associated rw factor Z_i of those items that at least one group member has rated *explicitely* by only using the explicit ratings of the users. We call these group ratings that are based on the individual group members' preferences *estimated group ratings*:

$$R_i = \frac{\sum_{n \in rated(i)} r_{n,i}}{\sum_{n \in rated(i)} 1} \tag{11}$$

$$Z_i = \sum_{n \in rated(n,i)} \frac{1}{|explicitRatings(n)|} \tag{12}$$

Equation 11 simply sets the average of the individual ratings as the new group rating of item i. Equation 15 takes the different numbers of explicit ratings by the group members into account. Imagine a group member who has only rated 5 items and another one who has rated 100 items. The latter member's preferences would have a much stronger impact on the final group recommendation without any type of normalization. This is done in equation 15 by weighting the group rating indirect proportionally to the number of explicit ratings of the group members whose preferences were averaged in order to create the group rating. Thus, a group rating that consists of the rating of a group member who has rated only 5 movies is weighted higher than the group rating that consists of the rating of a group member who has rated 100 movies.

Additionally, equation 15 weights estimated group ratings the higher the more individual ratings were used to create the estimated group rating. This is also desirable, because these ratings could be considered to mirror the group's real preferences better than a rating that is based on the preference of only one group member for example.

In the second step the algorithm includes again the genre information the same way it is done with the individual rating vectors by computing genre ratings $R_i \in genreGroupRatings$ by an averaging operation. It is important to mention that neither the previously generated group ratings $R_i \in estimatedGroupRatings$ nor the associated rw factors of the group ratings are touched. The only thing that is still missing now are the genre ratings' rw factors. They are computed as follows in order to end up with a group rating vector that has the same properties as an individual rating vector:

$$Z_i = \frac{1}{x} \cdot \frac{\sum_{R_j \in estimatedGroupRatings} z_j}{|genreGroupRatings|} \tag{13}$$

Importance of Individual Users A further consideration during the computation could be the higher importance of individual group members compared to the others. Imagine a birthday party, where all people want to watch a movie.

Probably the person who celebrates his or her birthday is considered very important during the group decision process.

This could be easily integrated in the algorithm described so far due to the rw factor modification. The rw factors of those group ratings resulting of the person's preferences just have to be increased by a facor that mirrors the person's importance. However, in our scenario, we assume that all group members are equally important. Nevertheless, the notion of more important persons can be used to implement some kind of fairness.

Consideration of Fairness As discussed in section 3.2 users care for fairness issues and it would probably be desirable if the average individual satisfaction after x recommendations would be approximately the same for all group members. If we think of a group which currently accepted a recommendation that dissatisfies one group member and we assume the group to be fair and caring for the satisfaction of all group members, than it is plausible to assume a shift in the group's preference model for the next recommendation in favor of the dissatisfied person.

This can be modeled by applying the average satisfaction $\overline{S_n}$. After each recommendation we recompute the group preference model as described above with modified equations 11 and 15:

$$R_i = \frac{\sum_{n \in rated(n,i)} r_{n,i} \cdot \frac{1}{\overline{S_n}}}{\sum_{n \in rated(n,i)} \frac{1}{\overline{S_n}}} \tag{14}$$

$$Z_i = \sum_{n \in rated(n,i)} \frac{1}{\overline{S_n} \cdot |explicitRatings(n)|} \tag{15}$$

The modification of equation 11 ensures that a dissatisfied (i.e. important) person's preferences are weighted higher when "averaging" the individual ratings. This also propagates later to the group genre ratings. Additionally, the modification of equation 15 increases the impact of a rating on the final group prediction, depending on the (dis)satisfaction of the people whose ratings were used to create the group rating.

Integration of Group Ratings The group members are able to rate movies as a group in our scenario, so there has to be a mechanism that transfers these group ratings into our group preference model. Without a doubt the explicit ratings by the group as a whole should be considered very valuable due to the fact that all other group ratings are only estimations of the group's preferences. In contrast, the explicit ratings by the group actually describe the real group's preferences.

If the group explicetely rates items, this results in an explicit group rating vector $\vec{g}$ with ratings g_i for those items $I_{explicitelyRated}$ where an explicit group rating exists. We define that the explicit ratings by the group should have an impact on the final group prediction that is y times higher than that one of the

other group ratings $I_{notExplicetlyRated}$, i.e. estimated group ratings and genre ratings. So our algorithm first constructs the group preference model by the two steps described above and then applies the explicit group ratings to the group rating vector in a third step. If there is an explicit group rating a possibly existent estimated rating is replaced of course.

$$R_i = g_i \ \forall i \in I_{explicitelyRated}$$

For every explicit group rating the rw factor is set as

$$Z_i = y \cdot \frac{\sum_{j \in I_{notExplicitelyRated}} z_j}{|I_{explictelyRated}|} \tag{16}$$

Moreover, the explicit group rating vector can be integrated in the item-item matrices, but this is technically not more than another user that helps the item-item matrix to describe item-item similarities more precisely.

Storage of the Group Preference Model Due to privacy concerns the individual user rating vectors of other users are not stored on an user's mobile device. Only the recommender model, i.e. the item-item matrix, and the own rating vector is stored.

In order to be able to transfer the approach of group preference models to a real decentral recommender system without the need of a stationary device that stores the data of all existent groups, at least one mobile device of the group has to store the explicit group rating vector which consists of the group's explicit ratings. This is necessary for the algorithm to be able to use these explicit ratings in the group recommendation process. If we also want to consider the fairness issue, than every group member should store its satisfaction with the previous recommendations regarding a specific group.

When storing these data on the mobile devices, there has to be an group identifier, so that a mobile device recognizes when a group has gathered in the vicinity, for which group information is available. With regard to privacy issues this could be done for example by a hash value that is generated by concatenating the group member's names (e.g. alphabetically) and encrypting them by means of a hash function.

It is possible that only a subset of a former bigger group gathers and wants the system to generate recommendations. In this case the system should be able to adapt the information for the complete group in order to get information that could be used to generate predictions for the subset. This would also require every group member to store the explicit group rating vector because at least one member of the subset must be able to provide the group rating vector to the recommendation algorithm. However, we will not cover this consideration in detail, due to the limited scope of this paper.

Summary of the three steps of the algorithm

1. Aggregate the group members' individual explicit ratings by averaging them. This results in estimated group ratings.

2. Compute the genre information for the group rating vector based upon the averaged estimated group ratings. Weight the genre ratings.

3. Integrate the explicit group ratings and weight them.

4.3 Aggregating Recommendations

After we have discussed the generation of a group preference model we want to present a second algorithm, which aggregates individual predictions in order to come up with a group recommendation. This approach was already shortly introduced in 3.4.2.

Aggregation Method Until the individual preference predictions have to be aggregated the algorithm does not differ from the procedure when the system makes a recommendation for a single user. The group members meet at the stationary device, which in this case is not much more than a "power-user", who has contact to potentially many more users than a normal single user is expected to have. After all, for every group member n the system computes a rating prediction $pred(n, i),\ i \in unrated(n)$ for all explicitely unrated items $i \in unrated(n)$.

We now follow the considerations stated in 3.4.1. Given the (partially predicted) preferences of all group members $n \in G,\ G \subseteq N$ we use the decision strategy *Average Without Misery* in order to rank the the recommendation candidates. So for an item i the *aggregated group rating* is computed as follows:

$$pred(i) = \frac{\sum_{n \in G} pred(n, i)}{|G|} \tag{17}$$

If there is at least one group member whose predicted preference for an item $pred(n, i)$ is below a threshold value λ, the item is dismissed. The candidate items with the highest aggregated group ratings could finally be chosen for recommendation.

Importance of Individual Users The considerations about important users discussed above at our first approach also apply here and the methods described below for fairness issues can be used for implementation of importance of certain users.

Consideration of Fairness In order to account for fairness in a sequence of recommendations we recompute the aggregated group ratings with a modified equation 17 by using the average satisfaction $\overline{S_n}$ of user n already introduces in our first approach:

$$pred(i) = \frac{\sum_{n \in G} pred(n, i) \cdot \frac{1}{\overline{S_n}}}{\sum_{n \in G} \frac{1}{\overline{S_n}}} \tag{18}$$

Integration of Group Ratings As described above, if the group explicitely rates items on a stationary device, this results in an explicit group rating vector. Of course, our second approach should also account for these explicit ratings. Thus, the algorithm also computes predictions $pred(g, i)$ for every item i based on the explicit group ratings and integrates them with the already aggregated group ratings $pred(i)$:

$$pred'(i) = \frac{pred(i) + \alpha \cdot pred(g, i)}{1 + \alpha} \tag{19}$$

By modifiying the factor α the impact of the explicit group ratings on the final group rating prediction can be controlled.

Storage of the Group Preference Model The same considerations as discussed with our first approach also apply with the second approach.

5 Evaluation

5.1 Prelimenary Considerations

Due to the limited scope of this work we were not able to evaluate all aspects of the former presented approaches, because several difficulties make the evaluation of group recommendation techniques a challenging and major task. The most difficult part is about measuring the satisfaction of a group with a particular recommendation or recommendation sequence. When dealing with usual single recommender systems it is an easy task to measure the satisfaction of an individual. An evaluator simply divides the given (explicit) ratings in a training set and a test set. Hence, the recommender system is trained with the ratings in the training set and tries to predict the ratings of the test set as precisely as possible. In this case the prediction for a particular item just has to be matched against the actual (explicit) rating of this item, because the user explicitely told the evaluator how satisfied he is with the item by means of his explicit rating. Because the evaluation compares predictions with ratings the user has already stated and no interaction with the user is needed, this approach is also called *offline analysis*.

However, this simple evaluation technique cannot be transferred to group recommendation systems without any further considerations. Imagine a group of people, who receive a movie recommendation. If the group wants to rate this evaluation, the first requirement would be that every member of the group has already seen the movie. Of course, the movie could be rated by the group with some members not having seen the movie, but this rating would be based on the subjective assumptions of the uninformed group members and it is likely that the actual group rating will change when all group members have actually seen the movie (e.g. after leaving the cinema).

Eventually, there are two possible ways to get ratings for a group as a whole, i.e. *group ratings*. The first alternative are *explicit group ratings*, which means

that the group members are aware of building a group and rate the movie together. In this case, only the former requirement (that all group members must have seen the movie) applies. The second alternative is the generation of a group rating out of the individual's ratings for a movie. Though, even with the ratings of the individual group members at hands the measure of the group's satisfaction, i.e. finding a group rating, is not trivial. Should the satisfaction be denoted as the simple average of the individuals' ratings or should the group rating decrease when the individual rating of at least one group member is below a certain threshold?

Things get really complicated if the evaluation tries to account additionally for different sequences of recommendations. Whereas a recommender for an individual should usually rank a list of suggested items simply by the prediction value, a group recommender should perhaps be able to care about fairness issues and look for items which nobody of the group really hates (although the general average rating might be high for that item).

In a nutshell, the group ratings are either given explicitely or individual ratings have to be aggregated by an *aggregation function*. Recommendations and especially sequences of recommendations should be benchmarked by some kind of *satisfaction function*. [16] tries to evaluate different satisfaction functions in a small user study.

5.2 Evaluation Design

We decided to conduct an offline analysis by using the MovieLens dataset provided by the GroupLens research project [24]. The dataset consists of 10 million ratings (on a discrete rating scale from 1 to 5) for 10681 movies by 71567 users. So there are no explicit groups respectively group ratings given by the dataset and have to be defined by ourselves. Of course a dataset that accounts for group recommendation would be a better choice, but generating such a dataset is far beyond the scope of this work and at the time of this writing we are not aware of any available dataset for such purposes. In order to collect data for such a dataset an online movie community with the ability of forming groups and rating movies would be very helpful ([15] presents efforts in that direction).

A second alternative would have been a live user study, but with respect to the scope of this work only a relatively small user study could have been conducted, which would had made it very difficult to present significant results and clarify results from various side effects. In particular, forming different groups of people out of the participants of a small user study, so that for all groups all group members have seen the same movies in order to be able to state group ratings (which then could be compared to group predictions) at all is a challenging task with a small amount of people. So the most promising way in our opinion is that of an online community like in [15, 24], that is designed for the evaluation of group recommendation techniques.

5.2.1 Research Questions

However, aside from the difficulty of finding a realistic aggregation and satisfaction function, the offline analysis with the MovieLens dataset allowed us to explore the following research questions:

- Which of our two presented approaches works best for group recommendation in a mobile scenario?
- Does the use of movie genre information enhance recommender quality?
- How do the different approaches perform with different numbers of users generating the item-item matrix?
- How do the different approaches perform with different group sizes?
- Does the use of movie genre information enhance the quality of the item-item matrix?

5.2.2 Evaluation Method

At first, groups had to be found in the MovieLens dataset, for which rating predictions should be computed. In order to simulate explicit group ratings we chose the simple average of the group members' ratings for a movie (aggregated group rating). Therefore, all group members must have rated the movie.

We searched for groups with 3, 5 and 10 group members and at least 15 movies that all group members have rated. For a particular group we denote the set of group members $G \subset N$ and the set of commonly rated items $I_{common} \subset I,\ |I_{common}| = 15$. For every group the set of commonly items I_{common} is split into the disjoint sets $I_{train} \subset I_{common}$ with $|I_{train}| = 8$ and $I_{test} \subset I_{common}$ with $|I_{test}| = 7$. While the training set I_{train} is available to the recommender algorithm for training purposes, the algorithm eventually tries to predict the group ratings $r_i = \frac{\sum_{u \in G} r_{u,i}}{|G|},\ i \in I_{test}$ of all movies in the test set I_{test}. This results in a prediction $pred_i$ for every movie in the test set.

5.2.3 Performance Measures

Two different performance measures are used in the evaluation:

Mean Average Error (MAE) The MAE is a standard performance measure in the evaluation of recommender systems (for more details see [2]). It measures the average deviation between the predicted group rating and the "true" (aggregated) group rating. In our case, the MAE for a group G is defined as follows:

$$MAE = \frac{\sum_{i \in I_{test}} |pred_i - r_i|}{|I_{test}|} \tag{20}$$

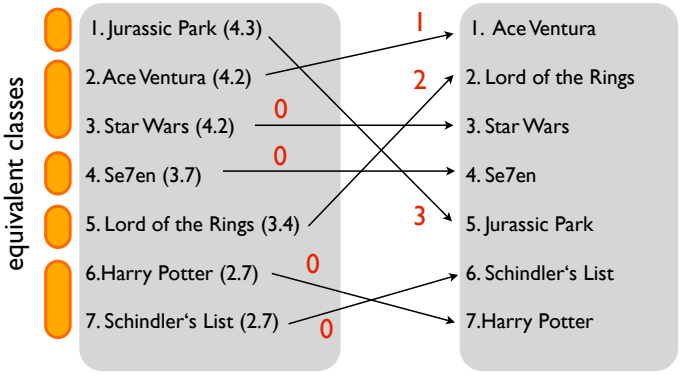


Figure 2: Rank performance measure

Rank Error (RE) Given the aggregated group ratings the items in the test set I_{test} can be ranked. Imagine a group of people who sit at home with seven DVDs available, but only time for watching two of them. So the recommender should be able to propose the two items with the highest group ratings. Therefore, we introduce a rank error (RE) that allows us to measure the ability of an algorithm to rank the items in the test set in the right order.

It is possible that the ranking of the items is ambiguous, e.g. two items could have an aggregated group rating of 4.2. All items that have the same aggregated group rating are in the same *equivalent class*.

The aggregated group ratings as well as the predictions result in a ranked item list. For every item $i \in I_{test}$ we denote $dist_i$ as the distance that item i is displaced in the prediction ranked list in contrast to its "true" ranking in the aggregated rating ranked list. Thereby the distance is only increased when traversing equivalent classes and not pure ranking places. So if the movies on place 2 and place 3 in the true ranking have an aggregated rating of 4.3 and the prediction list ranks the movie on place 2 in the true ranking on place 3, this would not increase the distance because the true ranking is ambiguous in this point.

An example is shown in figure 2. On the left a ranked list of the test items is shown, while on the right side a possible predicted ranked list is shown. The red numbers denote $dist_i$ for every item i.

Finally the RE is computed as follows:

$$RE = \sum_{i \in I_{test}} dist_i^2 \tag{21}$$

By squaring the distance values greater displacements are punished more than few minor place switches. Thus, regarding the example in figure 2, this would result in $RE = 1^2 + 2^2 + 3^3 = 14$.

5.2.4 Implemented Approaches

- *Random*
  For every rating prediction this algorithm simply computes a random float number between 1 and 5

- *Average*
  This algorithm takes the average rating for a movie by all users that were used to build up the item-item matrix

- *AP w/o rw factors*
  This is the most simple form of our first approach presented in section (4.2), without using the weighting factors introduced in equation(8) and without using any movie genre information when computing the group rating vector. m denotes the size of the similarity neighborhood used for computing the predictions.

- *AP with rw factors*
  This algorithm computes a group rating vector with considerations of equations(8),(11) and (15), but still without using any movie genre information when computing the group rating vector. m denotes the size of the similarity neighborhood used for computing the predictions.

- *AP with rw factors & genre information*
  This algorithm computes a group rating vector with considerations of equations(8),(11) and (15). In addition, movie genre information is considered as presented in equation (13). m denotes the size of the similarity neighborhood used for computing the predictions. After integrating movie genre information in rating vectors, the rating vectors tend to contain ratings for a great amount of movies. The algorithm for the generation of the similarity neighborhood would most likely result in a neighborhood that consists of almost m movies, that have genre ratings, because the chance is much higher, that the m movies with the highest similarity values are associated with a genre rating and not an explicit ratings. To overcome this we define the similarity neighborhood as the m movies associated with genre ratings in the rating vector and the highest similarity values *plus* the m movies associated with explicit ratings in the rating vector and the highest similarity values. So eventually the neighborhood contains $2m$ movies and the consideration of the explicit ratings is ensured.

- *simple AR*
  This is the most simple form of our second approach as presented in equation (17) without using any movie genre information in the individual rating vectors. m denotes the size of the similarity neighborhood used for computing the predictions.

- *AR with genre information*
  This algorithm is the same as *simple AR*, but with using movie genre information in the individual rating vectors (see equations (9) and (10)). m

denotes the size of the similarity neighborhood used for computing the predictions. See the explanation of *AP with rw factors & genre information* for details about the similarity neighborhood generation.

6 Discussion

We first tested the different algorithms presented in section 5.2.4 with 100 randomly chosen groups for each group size respectively (groupsize 3, 5 and 10) without considering movie genre information (see section 3.3) when computing the item-item similarity matrix. In order to examine how the different approaches perform with different numbers of users available for generating the item-item matrix, each algorithm was tested with 50, 1000 and 5000 random users building up the item-item matrix. This results in 9 different results for each algorithm, because there are 3 different groupsizes and 3 different numbers of users building up the item-item matrix. The results from this first experiment are shown in the figures 3 to 8.

Afterwards we picked some of the best performing algorithms and tested them again on 100 groups of groupsize 3 and 5 one time *without* genre information and the other time *with* genre information used for the computation of the similarity matrix in order to answer the question, whether the use of genre information in the item-item matrix enhances recommender quality or not. The results from this second experiment are shown in the figures 9 and 10.

With the results at hand we can now discuss the research questions stated in section 5.2.1:

Which of our two presented approaches works best for group recommendation in a mobile scenario with regard to different numbers of users and groupsizes? One of the most surprising findings is the high performance of the *Average* algorithm. The average of all ratings from all users (but only those that are also available for the generation of the item-item matrix) seems to be a pretty good way to predict the ranking of the 7 test movies. For all group sizes it clearly beats all other more complicated algorithms if there are only 50 users that are available to build an item-item matrix and the best performing algorithms are not significantly better with 1000 users. Only with 5000 users a significant difference can be found. This is plausible, because the item-item matrix represents the "brainpower" of the more complicated algorithms. 50 users are obviously not enough to build up a useful source of information in form of an item-item matrix. The more users are available to compute the item-item matrix, the more intelligent the more complicated algorithms get. This is also the reason why all more complicated algorithms aside from a few exceptions get clearly better the more users are used for the item-item matrix. So we put on record, that it depends on the expected number of contacts between the peers in a mobile scenario which algorithm is recommended. If the designers of a system expect the users to meet infrequently and not very often, i.e. only few users are available for the item-item matrix, then the use of an simple *Average*

Figure 3: Results of experiment 1: average RE with groupsize 3

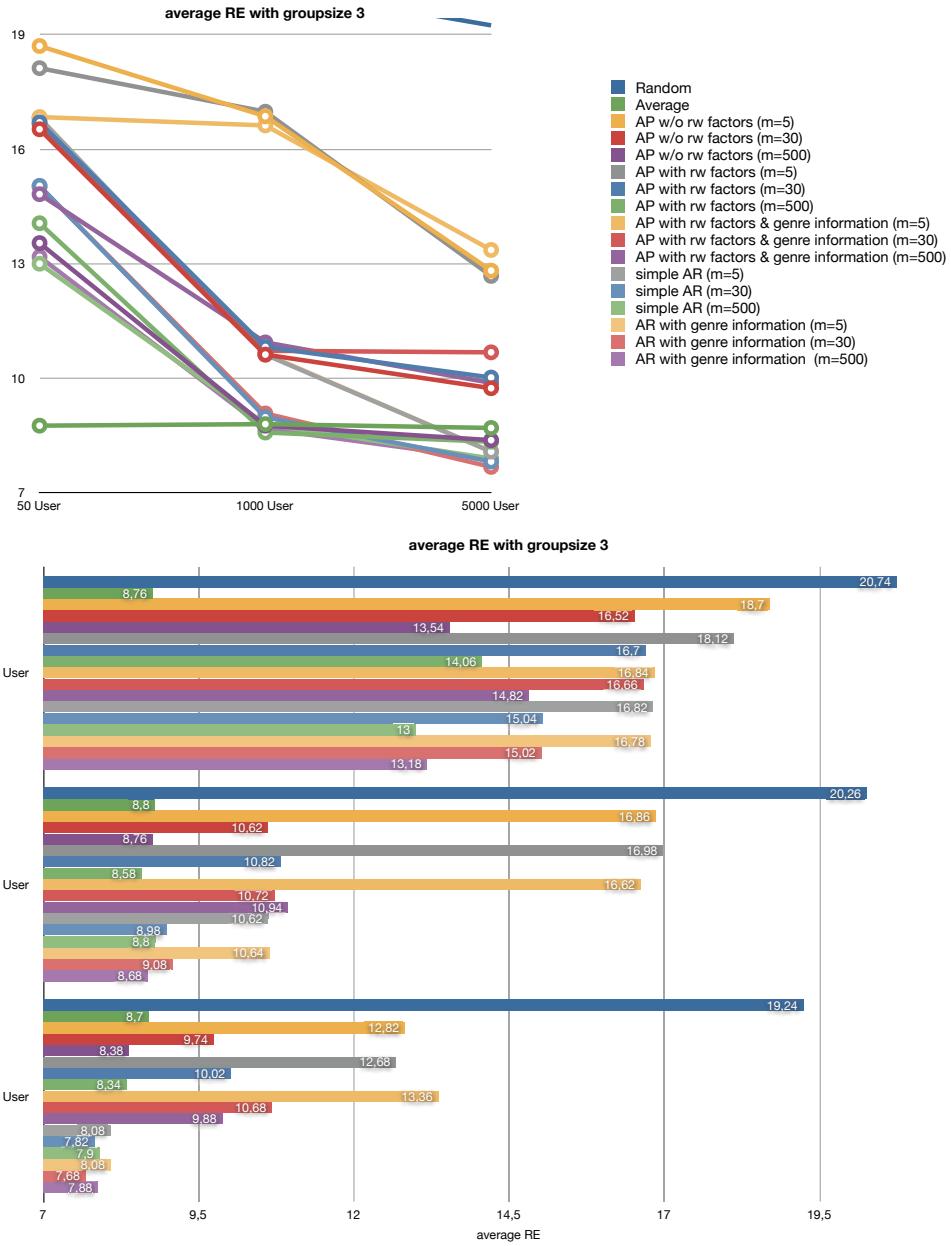


Figure 4: Results of experiment 1: MAE with groupsize 3

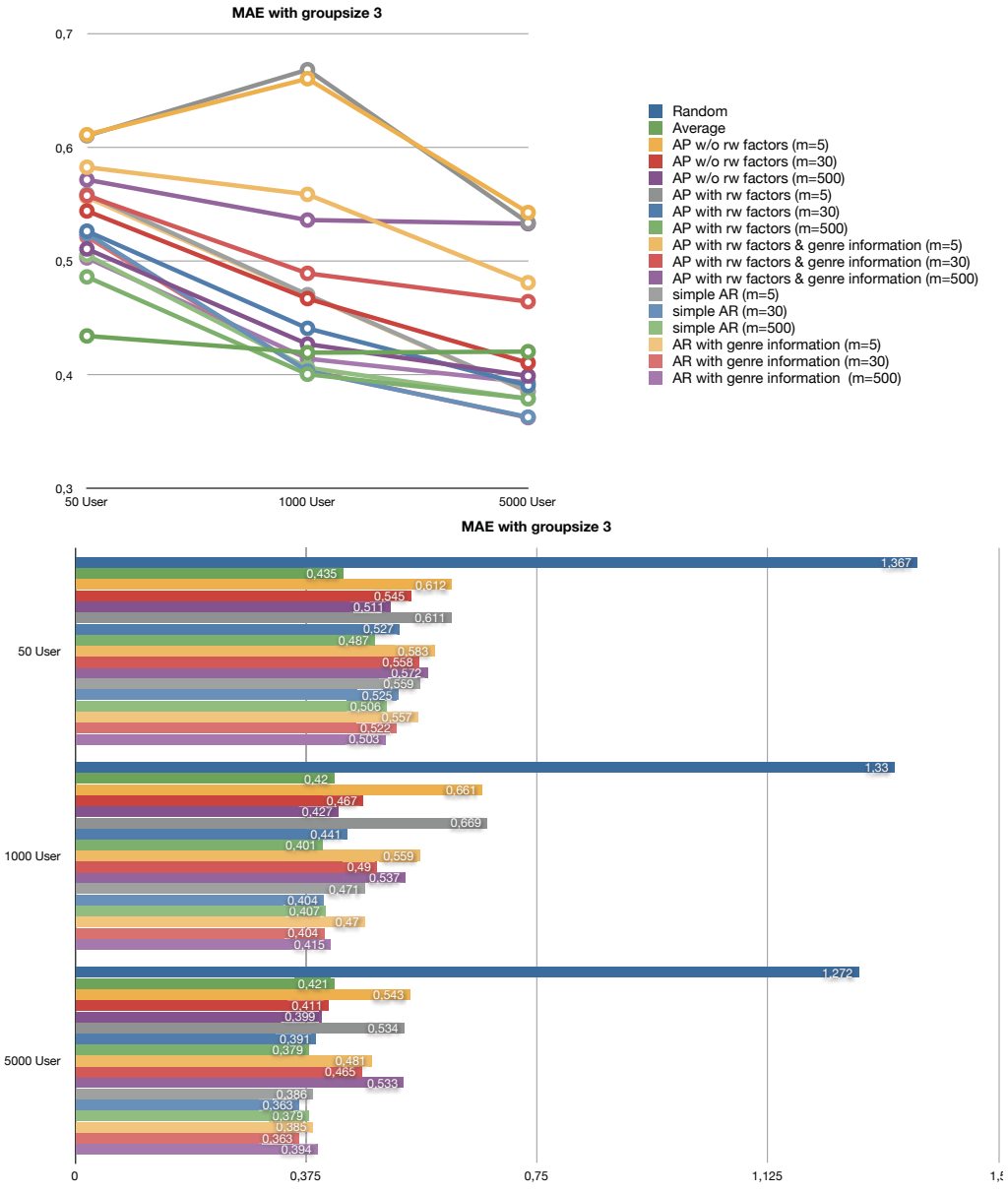


Figure 5: Results of experiment 1: average RE with groupsize 5

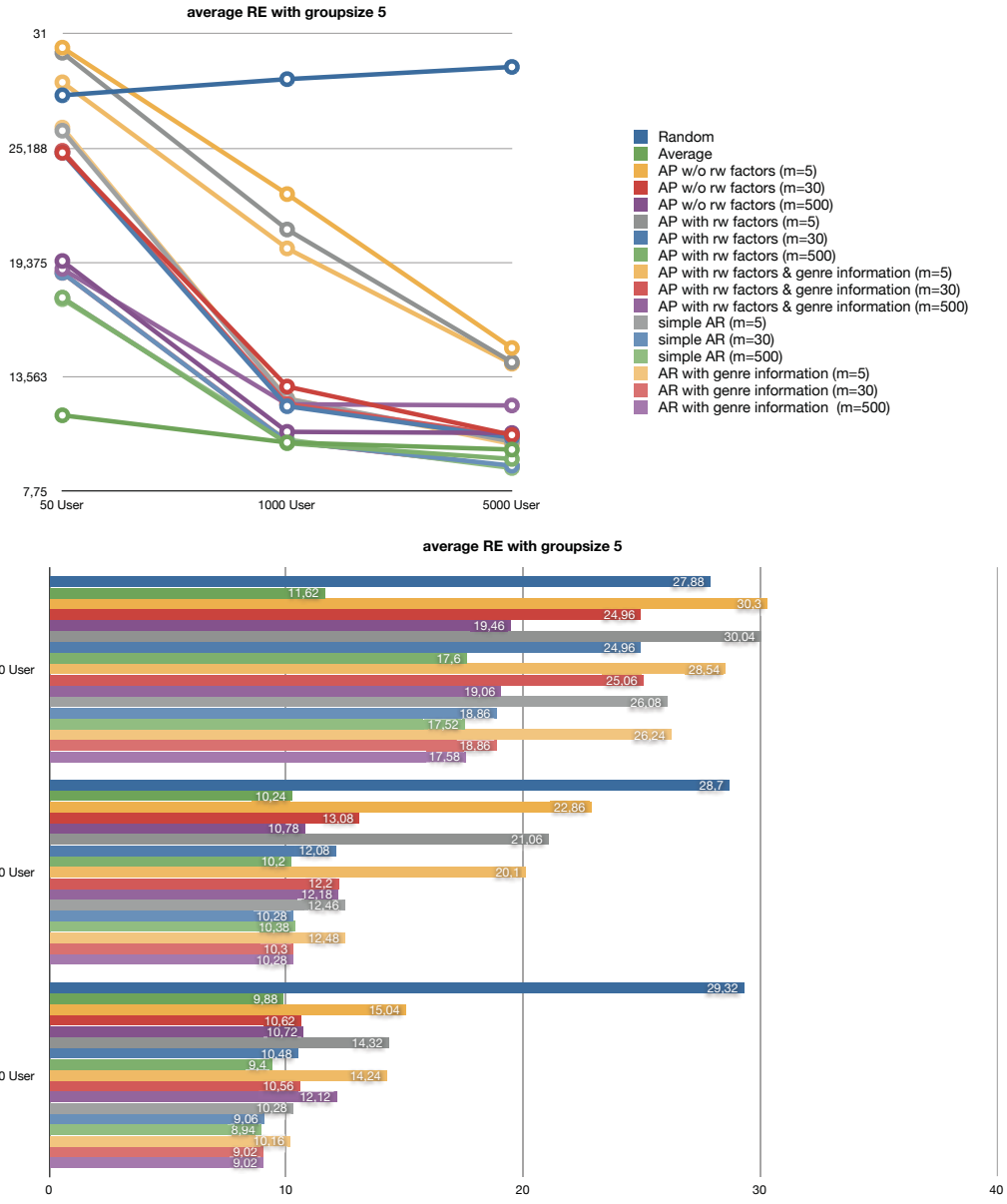


Figure 6: Results of experiment 1: MAE with groupsize 5

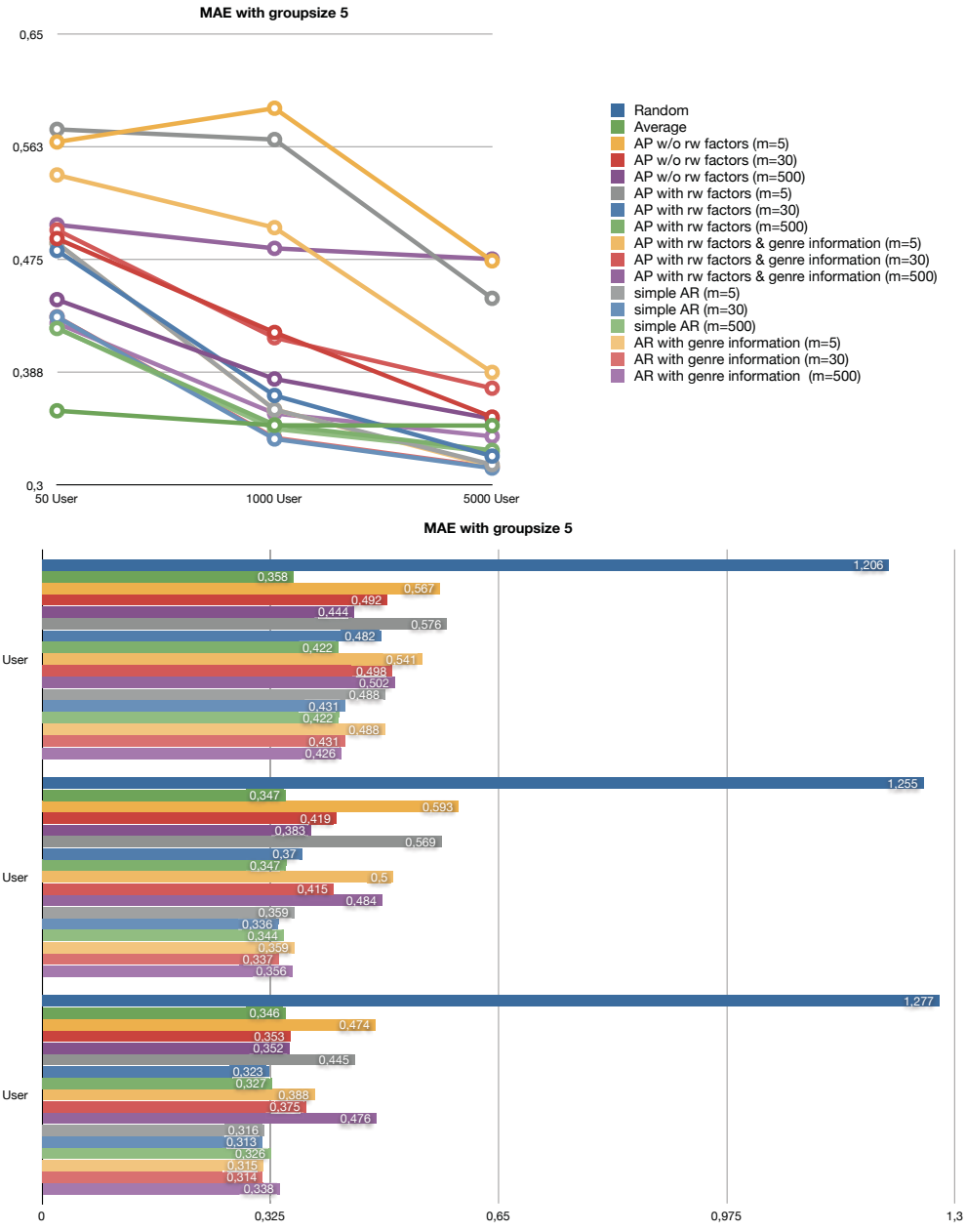


Figure 7: Results of experiment 1: average RE with groupsize 10

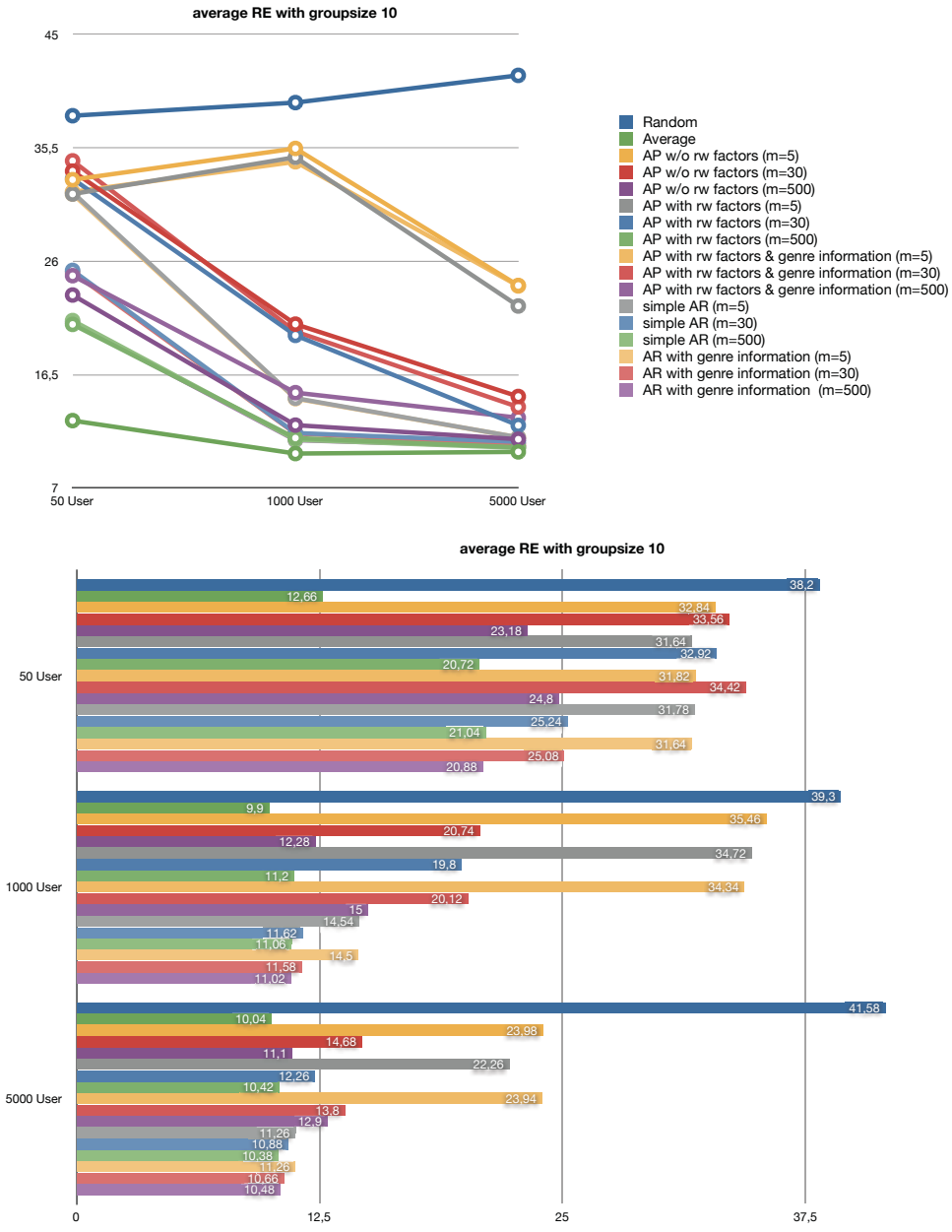


Figure 8: Results of experiment 1: MAE with groupsize 10

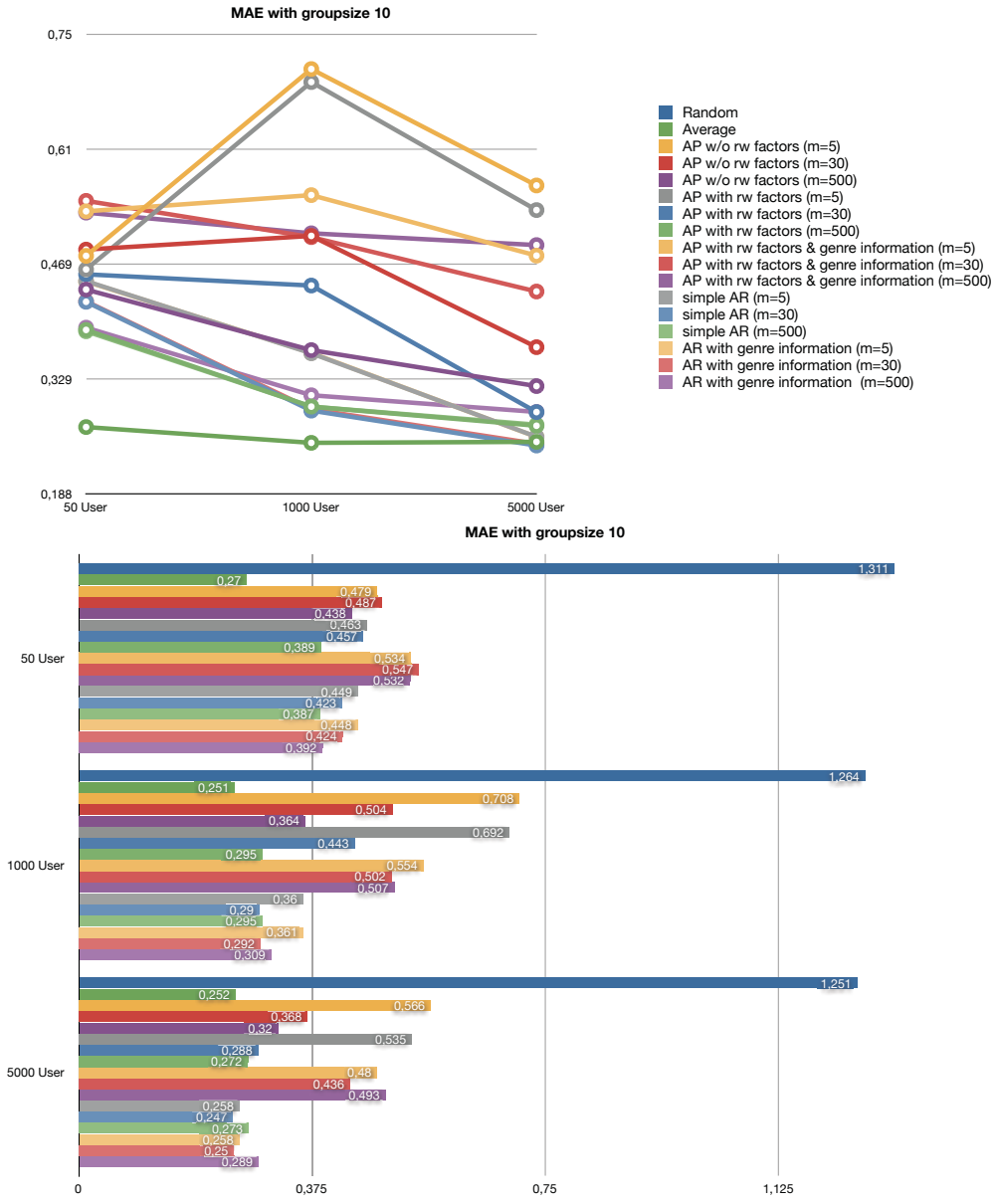


Figure 9: Results of experiment 2: genre information in item-item matrix (group size 3). Algorithms that use genre information are denoted by a star *

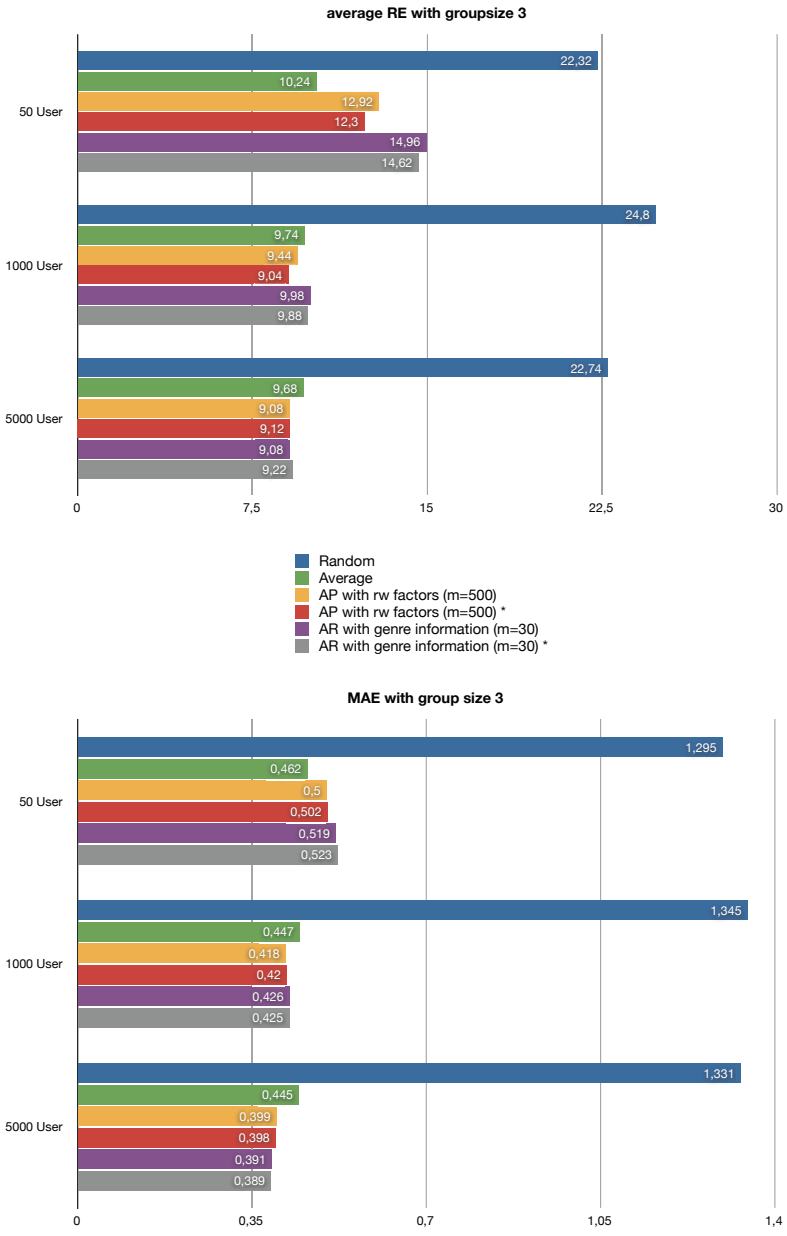
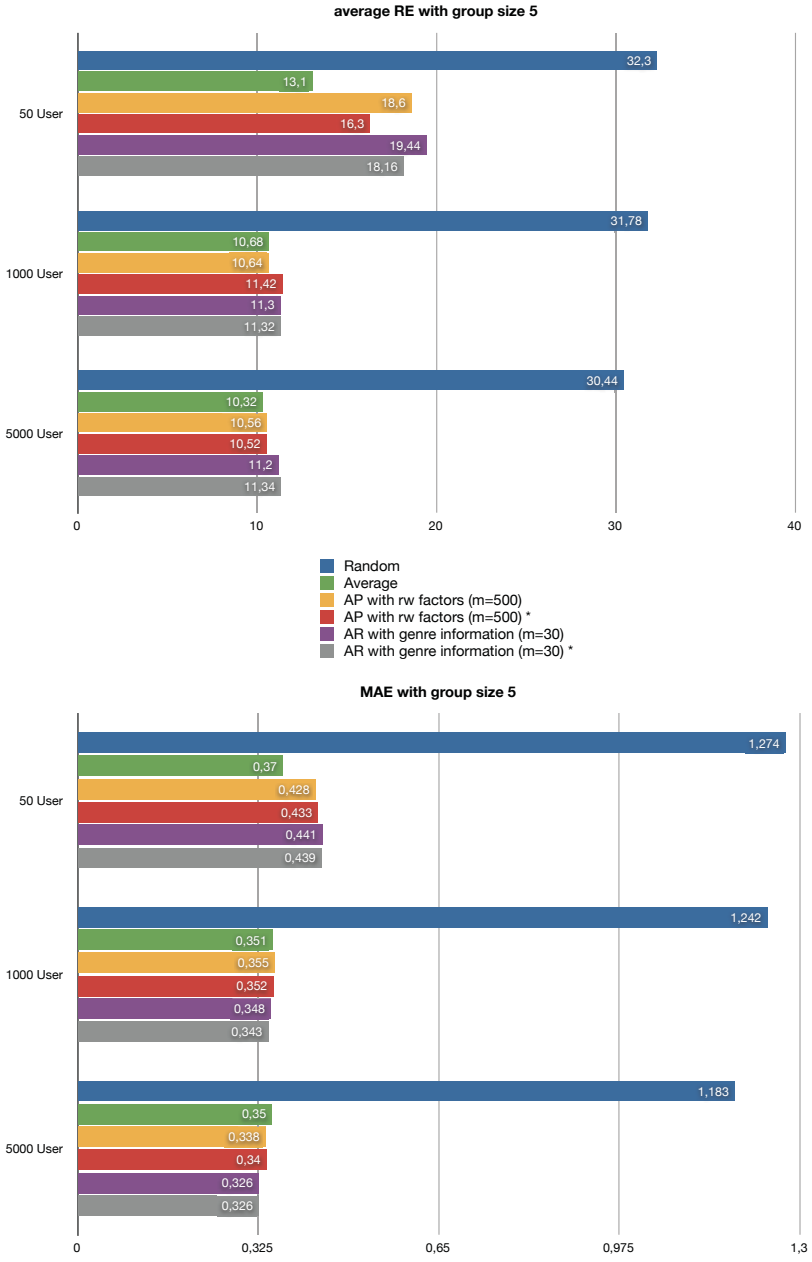


Figure 10: Results of experiment 2: genre information in item-item matrix (group size 5). Algorithms that use genre information are denoted by a star *



algorithm is the best choice by far. On the contrary, if the designers expect a very frequent exchange of rating vectors, e.g. because group recommendation takes place at a stationary device that has contact to many users, than a more complicated algorithm is the better choice over a simple *Average* algorithm.

On the other hand there can be seen an exception to this (apparently plausible) rule: at group size 10, the *Average* algorithm beats all other algorithms. Perhaps this is because the more group members there are, the more the aggregated group rating tends to be equal to the average rating for this item by all users. This does not have to be the case if other group rating metrics are applied or explicit group ratings are available, but the evaluation of this assumption is beyond the scope of this evaluation.

Aside from the results with group size 10, all *Aggregating Recommendations* algorithms perform pretty well, especially the variants with $m = 30$. Also *AP w/o rw factors (m=500)* and *AP with rw factors (m=500)* perform not really significantly worse.

So eventually, according to our results here, we summarize our design recommendations as follows:

- if only infrequent exchange of rating vectors between the peers is expected, i.e. few users are available for the item-item matrix, then a simple *Average* algorithm is recommended
- if rather big groups are expected (e.g. more than 10 group members), then a simple *Average* algorithm also probably beats the other algorithms (especially when considering speed)
- if rather small group sizes and frequent exchanges between the peers are expected than one of the *Aggregating Recommendations* algorithms would be a good choice

Does the use of movie genre information enhance recommender quality? As we can see throughout figure 3 to 8 the best variant of *AP with rw factors & genre information* for all group sizes and all number of users is worse or in some cases at least not significantly better than the best variant of *AP w/o rw factors* and *AP with rw factors* regarding the average RE and the MAE. The behaviour of *AR with genre information* related to *simple AR* is similar. The algorithm that uses the genre information is neither significantly worse nor significantly better with respect to the average RE and the MAE than the algorithm that do not use genre information.

We therefore conclude, that the use of genre information as proposed in equation 6 in individual rating vectors as well as in the group rating vectors during the prediction computation process does not significantly enhance recommender quality. A possible explanation for this finding could be, that the explicit ratings in a rating vector are far more meaningful than the proposed genre ratings and the number of explicit ratings for a typical user of an online movie recommender system like MovieLens suffices for a good recommender quality. Another expla-

nation would be, that the proposed genre ratings themselves are just not good predictors of the user's likes and dislikes.

Does the use of movie genre information enhance the quality of the item-item matrix? To examine this question we conducted a second experiment with groups of groupsize 3 and 5. Two of the best performing algorithms, *AP with rw factors (m=500)* and *AR with genre information (m=30)*, were tested on 100 groups, one time with a "usual" item-item matrix and the other time with an item-item matrix that uses rating vectors for which genre ratings have been generated before. *Random* and *Average* were also tested for comparison purposes.

The results are shown in figures 9 and 10 (algorithms that use genre information for computing the item-item matrix are denoted with a star *). However, as the charts show, both with the average RE and the MAE, there are no significant differences between the different methods for generating the item-item matrix. Only with 50 users there seems to be a slight advantage to the item-item matrix with movie genre information included, but this small advantage only shows at the RE metric and not the MAE metric. So we put on record, that at least with our method of integration of movie genre information in the item-item matrix, no relevant improvement is identifiable when using movie genre information except for a possible small improvement when dealing with only a few users building up the item-item matrix.

7 Conclusion

Finally we want to give a short summary of the work done in this paper. After a few introductory words we gave a very brief introduction to the field of usual recommender systems. Following that we looked at the field of group recommender systems and listed the various difficulties and issues one has to bear in mind when dealing with recommendations for more than one single active user. Our concrete scenario was a movie recommender in a mobile, decentralized scenario with a stationary device, which allows groups of people to receive recommendations and input explicit ratings for movies for the group as a whole. We followed the implementation approach in [13] and therefore limited ourselves to the use of an model-based approach with an item-item matrix due to privacy and memory reasons.

We then presented two different approaches for group recommendation with regard to all issues, that were stated already above when introducing group recommender systems. In particular we took into account different types of ratings (estimated aggregated group ratings, genre ratings and explicit group ratings) and therefore proposed an extension to the usual rating vector datastructure by introducing weighting factors. We showed also, how it would be possible to care for fairness in a sequence of recommendations for a group and shortly discussed storage issues of such group datastructures in a decentralized scenario with mobile devices.

Eventually, we conducted an offline analysis of the presented approaches by splitting them up into several algorithms and test them for two different performance measures on different groupsizes and different numbers of users building up the used item-item matrix. We took the MovieLens dataset as evaluation database. Several previously stated research questions could be answered by this first experimental run. Afterwards, we took some of the best performing algorithms and did a comparative test with an item-item matrix that put into account genre information and an item-item matrix that did not, in order to examine the effects of the use of genre information in the generation of the item-item matrix.

The results showed a strong performing simple *Average* algorithm, which we recommend to use when expecting only few rating vector exchanges between the participating peers. In the other case, the *Aggregating Recommendations* algorithms did perform pretty well and are recommended to use. Genre information does not seem to significantly enhance the quality of the item-item matrix and, according to our results, does not justify the higher memory needs through higher quality.

Due to the limited scope of this work only an offline analysis with the MovieLens dataset could be conducted. The several not trivial difficulties that occur with a group recommender evaluation demand for an extensive dataset, which is explicitly designed for the evaluation of group recommender techniques. To our knowledge, this would be best done with the data of an online recommender system. The precise evaluation of the here presented approaches with regard to the many possibly distorting side effects could be the subject of a future research project.

References

[1] G. Linden, B. Smith, J. York, 2003. *Amazon.com Recommendation: Item-to-Item Col laborative Filtering.* IEEE Internet Computing.

[2] J. L. Herlocker, J. A. Konstan, L. G. Terveen, J. T. Riedl, 2004. *Evaluating collaborative filtering recommender systems.* ACM Trans. Inf. Syst. 22(1), 5–53.

[3] G. Adomavicius, A. Tuzhilin, 2005. *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions.* IEEE Transactions on Knowledge and Data Engineering, Archive Volume 17 , Issue 6, pp. 734 - 749.

[4] P. Resnick, H. R. Varian, 1997. *Recommender Systems.* Communications of the ACM, Archive Volume 40 , Issue 3, Pages: 56 - 58.

[5] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, 1997. *GroupLens: applying collaborative filtering to Usenet news.* Communications of the ACM, - eprints.kfupm.edu.sa.

[6] G. Salton, M. McGill, 1983. *Introduction to Modern Information Retrieval.* McGraw- Hill, New York.

[7] M. J. Pazzani, D. Billsus, 2007. *Content-Based Recommendation Systems. The Adaptive Web: Methods and Strategies of Web.* Springer-Verlag Berlin Heidelberg, pp. 325 – 341.

[8] J. B. Schafer, D. Frankowski, J. Herlocker, S. Sen, 2007. *Collaborative Filtering Recommender Systems.* Springer Berlin / Heidelberg, Volume 4321/2007, The Adaptive Web.

[9] P. Resnick, N. Iacovou, M. Suchak, P. Berstrom, J. Riedl, 1994. *Grouplens: An open Architecture For Collaborative Filtering of Netnews.* Proceedings of the 1994 ACM conference on Computer supported cooperative work, Chapel Hill, North Carolina. ACM Press pp. 175-186.

[10] J. S. Breese, D. Heckerman, C. Kadie, 1998. *Empirical Analysis of Predictive Algorithms for Collaborative Filtering.* Proceeding of the Fourteenth Conference on Un- certainty in Artificial Intelligence (UAI). Madison, Wisconsin. Morgan Kaufmann pp. 43-52.

[11] J. L. Herlocker, J. A. Konstan, J. Riedl, 2000. *Explaining collaborative filtering recommendations.* Proceedings of the 2000 ACM conference on Computer supported cooperative work.

[12] J. McCarthy, T. Anagnost, 1998. *MusicFX: An arbiter of group preferences for computer supported collaborative workouts.* Proceedings of the 1998 Conference on Computer- Supported Cooperative Work. (1998) pp. 363–372.

[13] W. Wörndl, H. Mühe, V. Prinz, 2009. *Decentral Item-based Collaborative Filtering for Recommending Images on Mobile Devices.* Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, pp. 608-613.

[14] B. N. Miller, J. A. Konstan, J. T. Riedl, 2004. *PocketLens: Toward a personal recommender system.* ACM Transactions on Information Systems, vol. 22, no. 3, pp. 437–476.

[15] M. O'Connor, D. Cosley, J. A. Konstan, J. Riedl. *PolyLens: A recommender system for groups of users.* Proceedings of the European Conference on Computer . . . , 2001 - cs.umn.edu.

[16] J. Masthoff, 2004. *Group modeling: Selecting a sequence of television items to suit a group of viewers.* User Modeling and User-Adapted Interaction, Springer Verlag.

[17] A. Jameson, B. Smyth, 2007. *Recommendation to groups.* Lecture Notes in Computer Science, Springer Verlag.

[18] A. Crossen, J. Budzik, K. Hammond, 2002. *Flytrap: Intelligent group music recommendation.* In Gil, Y., Leake, D., eds.: IUI 2002: International Conference on Intelligent User Interfaces. ACM, New York (2002) 184–185.

[19] D. M. Pennock, E. Horvitz, C. L. Giles, 2000. *Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering.* In Proceedings of the Seventeenth National Conference on Artificial Intelligence.

[20] K. J. Arrow, 1950. *A difficulty in the concept of social welfare* - The Journal of Political Economy.

[21] H. Lieberman, N. Van Dyke, A. Vivacqua, 1999. *Let's browse: a collaborative browsing agent.* Knowledge-Based Systems - Elsevier.

[22] L. M. de Campos, J. M. Fernández, L. J. F. Huete, M. A. Rueda-Morales, 2008. *Managing uncertainty in group recommending processes.* Springer Science+Business Media B.V.

[23] A. Jameson, T. Kleinbauer, 2004. *Two methods for enhancing mutual awareness in a group recommender system.* Proceedings of the working conference on Advanced visual interfaces, pp: 447 - 449.

[24] *GroupLens internet ressource.* http://grouplens.org/, 2009