Rating Elicitation Strategies for Collaborative Filtering

Mehdi Elahi, Valdemaras Repsys, and Francesco Ricci

Free University of Bozen-Bolzano,
Piazza Domenicani 3, Bozen-Bolzano, Italy
mehdi.elahi@stud-inf.unibz.it
valdemaras@gmail.com
fricci@unibz.it
http://www.unibz.it

Abstract. The accuracy of collaborative filtering recommender systems largely depends on two factors: the quality of the recommendation algorithm and the nature of the available item ratings. In general, the more ratings are elicited from the users, the more effective the recommendations are. However, not all the ratings are equally useful and therefore, in order to minimize the users' rating effort, only some of them should be requested or acquired. In this paper we consider several rating elicitation strategies and we evaluate their system utility, i.e., how the overall behavior of the system changes when these new ratings are added. We simulate the limited knowledge of users, i.e., not all the rating requests of the system are satisfied by the users, and we compare the capability of the considered strategies in requesting ratings for items that the user experienced. We show that different strategies can improve different aspects of the recommendation quality with respect to several metrics (MAE, precision, ranking quality and coverage) and we introduce a voting-based strategy that can achieve an excellent overall performance.

Keywords: Recommender Systems, Active Learning, Rating Elicitation.

1 Introduction

Recommender Systems (RSs) support users in choosing the right products or services to consume by providing personalized suggestions that match the user's needs and constraints [11]. In this paper we are concerned with collaborative filtering (CF) RSs [5]; these systems use item ratings provided by a population of users to predict unknown ratings of the current user, and recommend the items with the largest predicted ratings. CF rating prediction accuracy does depend on the characteristics of the prediction algorithm, but also on the ratings known by the system. The more (informative) ratings are available the higher the recommendation accuracy is. In fact, in [10] it is shown that the recommendation accuracy can be improved to a larger extent if the ratings are acquired with a well designed selection strategy compared with the "classic" strategy where the users self-select the items to rate.

C. Huemer and T. Setzer (Eds.): EC-Web 2011, LNBIP 85, pp. 160–171, 2011.

Rating elicitation has been also tackled in some previous research works [8,9,1,4,3] but these papers focused on a different problem, namely the benefit of rating elicitation for a single user, e.g., in the sign up stage. Conversely, we consider the impact of several (some original) elicitation strategies on the system overall effectiveness (more details are provided in Section 5). We measured their effect using several evaluation metrics, including: the rating prediction accuracy (Mean Absolute Error), the number of acquired ratings, the recommendation precision, the system coverage, and the effectiveness of the recommendations' ranking, measured with normalized discounted cumulative gain (NDCG).

Moreover, we explore another new aspect, i.e., the performance of an elicitation strategy taking into account the size of the rating database, and we show that different strategies can improve different aspects of the recommendation quality at different stages of the rating database development. In this context, we have verified an hypothesis made originally by [9], i.e., that certain strategies, for instance, requesting users to rate the items with the largest predicted ratings, may generate, a system-wide bias, i.e., they can increase, rather than decrease, the system error.

In order to perform such an evaluation, we have created a system which simulates a real process of rating elicitation in a community of users, the consequent rating database growth starting from a relatively small set of data (cold-start), and the system adaptation (retraining) to the new data. In these simulations we used a state of the art Matrix Factorization recommender algorithm [5]; so that the results here presented can provide useful guidelines for managing real operational RSs.

In conclusion in this paper we provide a realistic, comprehensive evaluation of several, applicable and novel, rating elicitation strategies, providing guidelines and conclusions that would help their exploitations in real RSs. This is an important and necessary preliminary step for the application of any rating elicitation strategy in a real operational and possibly conversational system; having the goal to reduce the effort spent by users in rating (unnecessary) items and to improve the quality of the recommendations for all. We note that in this paper we extend a previous work [2] by describing and evaluating more strategies, including the voting one, and evaluating their behaviors on larger and a more realistic data set.

The rest of the paper is structured as follow. In section 2 we introduce the rating elicitation strategies that we have analyzed, and in section 3 we present the simulation procedure that we designed to evaluate their effects. The results of our experiments are shown in section 4. In section 5 we review some related research, and finally in section 6 we summarize the results of this research and we outline some future work.

2 Elicitation Strategies

A rating dataset R is an $n \times m$ matrix of real values (ratings) with possible null entries. The variable r_{ui}, denotes the entry of the matrix in position (u, i), and

contains the rating assigned by the user u to the item i. r_{ui} can store a null value representing the fact that the system does not know yet the opinion of the user on that item. In the Movielens dataset, which was used in our experiments, ratings are integers between 1 and 5 included. A rating elicitation strategy S is a function $S(u, N, K, C_u) = L$ which returns a list of $M \leq N$ items $L = \{i_1, \ldots, i_M\}$ whose ratings should be asked to the user u, where N is the maximum number of ratings to be elicited. K is the $n \times m$ matrix containing the known ratings, in other words, the ratings (of all the users) that have been already acquired by the RS. Finally, C_u is the set of candidate items whose ratings have not yet been asked to u, hence potentially interesting to be acquired. In fact, a strategy must not ask a user to rate the same item twice, i.e., the items in L must be removed from C_u.

Every strategy we propose analyzes the dataset of known ratings K and assigns a score to each item in C_u measuring how valuable it is to acquire the user opinion for that item. Then the N items with the highest score are identified, if the strategy can compute N scores, otherwise a smaller number of requests (M) is returned. Then, these items are actually presented to the user u to provide his ratings. It is important to note that the user may not have experienced some of these items; in this case the system will obtain less ratings.

We have considered many strategies; the first three below have been reported previously, while the rest are either original or have not been tested previously.

Popularity: the score for the item i is equal for all the users, and it is the number of not null ratings for i in K, i.e., those already acquired by the system. More popular items are more likely to be known by the users, and hence it is more likely that a request for such a rating will really expand the rating database [1] [9].

log(popularity)∗entropy: the score for the item i is computed by multiplying the logarithm of the popularity of i with the entropy of the ratings for i in K. This strategy tries to combine the effect of the popularity score, which is discussed above, with the heuristics that items with more diverse ratings (larger entropy) bring more useful information about the user preferences [1] [9].

Binary Prediction: the matrix K is transformed into a matrix B with the same number of rows and columns, by mapping null entries in K to 0, and not null entries to 1. A factor model is built using B as training data and then the prediction $\hat{b}_{ui}$ for each item i in C_u is computed and assigned as the score for the item. This strategy tries to predict what items the user has experienced, to maximize the probability that the requested ratings could be added to K (similarly to the popularity strategy).

Highest Predicted: a rating prediction $\hat{r}_{ui}$ is computed for all the items i in C_u (using the ratings in K) and the score for i is set to this predicted rating $\hat{r}_{ui}$. The idea is that the best recommendations could also be more likely to have been experienced by the user and their ratings could also reveal useful information on what the user likes. This is the default strategy for RSs, i.e., enabling the user to rate the recommendations.

Lowest Predicted: for each item in the C_u a rating prediction $\hat{r}_{ui}$ is computed (using the ratings in K). Then the score for item i is $M_r - \hat{r}_{ui}$, where M_r is the maximum rating value (e.g., 5). Lowest predicted items are likely to reveal what the user does not like, but should actually collect a few ratings, since the user is unlikely to have experienced all the items that he does not like.

Highest and Lowest Predicted: for each item i in C_u a prediction $\hat{r}_{ui}$ is computed (using the set of ratings in K). Then the score for i is $|\frac{M_r - m_r}{2} + m_r - \hat{r}_{ui}|$, where M_r (m_r) is the maximum (minimum) rating value. This strategy tries to ask for information on items that the user may or may not like.

Random: the score for an item is a random number. This is just a baseline strategy, used for comparison.

Voting: the score for the item i is the number of votes given by a committee of strategies including *popularity, variance* [12] *, entropy* [9]*, highest-lowest predicted, binary prediction,* and *random*. Each of these strategies produces its top 100 candidates for rating elicitation, and then the items appearing more often in these lists are selected. This strategy depends on the selected voting strategies, and we included random to impose an exploratory behavior that should improve the system coverage.

Finally, we would like to note that we have also evaluated other strategies: *variance, entropy*, and $log(pop) * variance$. But, since their observed behaviors are very similar to some of the previously mentioned strategies, due to lack of space they are not described here.

3 Evaluation Approach

In order to study the effect of the considered elicitation strategies we set up a simulation procedure. The goal was to simulate the evolution of the RS's performance exploiting these strategies. In order to run such simulations we partition (more details on the partition method are given later) all the available (not null) rating data in R into three different matrices with the same number of rows and columns as R:

- K: contains the ratings that are considered to be known by the system at a certain point in time.
- X: contains the ratings that are considered to be known by the users but not by the system. These ratings are incrementally elicited, i.e., they are transferred into K if the system asks for them from the (simulated) users.
- T: contains the ratings that are never elicited and are used only to test the strategy, i.e., to estimate the evaluation measures (defined later).

We also note that if $i \in C_u$ then its rating is worth acquiring because "unclear" to the system and candidate for elicitation, i.e., k_{ui} is null and the system has

not yet asked for this rating from u. That request may end up with a new (not null) rating $k_{ui} = x_{ui}$ inserted into K, if the user has experienced it, which is simulated by the fact that x_{ui} is not null in X, or in a no action, if this rating is not found in X. The system, in any case will remove that item from C_u, i.e., will not try to collect the same rating twice. It is important to note that in real scenarios the system may ask later on for a rating that the user is unable to provide at a certain point in time: because he may have experienced that item after the first request. This case is not considered in our simulation. Moreover, we observe that these three matrices partition the full dataset R: if r_{ui} has a not null value then either k_{ui} or x_{ui} or t_{ui} has that value, and only one of them is not null. The testing of a strategy S proceeds in the following way:

1. The not null ratings in R are partitioned into the three matrices K, X, T.
2. MAE, Precision, Coverage and NDCG are measured on T, training the prediction model on K.
3. For each user u:
   (a) Only the first time that this step is executed, C_u, the candidate set of user u is initialized to all the items i such that k_{ui} is null in K.
   (b) Using the strategy S a set of items $L = S(u, N, K, C_u)$ is computed.
   (c) L_e, which contains only items from L that have not null rating in X is created.
   (d) Assign to the corresponding entries in K the ratings for items in L_e as found in X and remove them from X.
   (e) Remove the items in L from C_u: $C_u = C_u \setminus L$.
4. Train the prediction model on K and compute MAE, Precision, Coverage and NDCG on T.
5. Repeat steps 3-4 (Iteration) for I times.

The MovieLens rating database were used for our experiments. Movielens consists of 1,000,000 ratings from 6,040 users on 3,900 movies. The experiments were conducted partitioning (randomly) the 1,000,000 not null ratings in the data set R in the following way: 2000 in K (i.e., very limited knowledge at the beginning), 698,000 in X, and 300,000 in T. Moreover, $|L| = N = 10$, i.e., the system at each iteration asks a simulated user for his ratings on 10 items. The number of iterations was $I = 200$, and the number of factors in the SVD prediction model was set to 16. It should be noted that we have also experimented with a denser initial matrix K containing 20,000 ratings. But, in spite of this difference similar results, as discussed below, were obtained.

When deciding how to split the available data into the three matrices K, X and T an obvious alternative choice was to respect the time evolution of the dataset, i.e., to insert into K the first 2000 ratings acquired by the system, then to use the second temporal segment of 698,000 ratings to populate X and finally to use the remaining ratings for T. Actually, it is not significant to test the performance of the proposed strategies for a *particular* evolution of the rating dataset.

Since we want to study the evolution of a rating data set under the application of a new strategy we cannot test it only against the temporal distribution of the data that was generated by a particular (unknown) previously used elicitation strategy. Hence we followed the approach also used in [3], i.e., to random split the rating data but we generated some (5) random splits of the ratings into K, X and T, and averaged the results. Besides, in this way we were able to generate users and items that had no ratings initially in the known dataset K. We believe this approach provided us with a realistic and hard experimental setup, allowing us to address the new user and new item problems [11]. In any case, additionally we performed the same experiments with the data partitioned by the natural order of acquisition time. The results were very similar to those observed in the random partitioning, confirming that the partitioning method does not impose any significant bias on the experiments.

We have considered four evaluation measures: mean absolute error (MAE), precision, coverage and normalized discounted cumulative gain (NDCG). Precision is computed considering, for each user, the top 10 recommended items (whose rating value appear in T) and judging relevant the items with ratings (in T) equal to 4 or 5. The coverage is measured as the proportion of the full set of items over which the system can form predictions or make recommendations [11]. Normalized Discounted Cumulative Gain (DCG) is a measure originally used to evaluate the effectiveness of information retrieval systems [7], but it is now becoming popular in RSs as well [13] [6]. NDCG measures the quality of a ranking comparing it to the best attainable one, i.e., the ranking where the recommendations are ordered in decreasing value of their actual ratings.

4 Experimental Results

4.1 Mean Absolute Error

MAE computed on the test ratings in T at successive iterations of the application of the considered elicitation strategies is depicted in Figure 1. Excluding the voting strategy, which needs particular discussion, there are two clearly distinct groups of strategies:

1. Strategies monotonically decreasing the error: lowest predicted, lowest-highest predicted, and random.
2. Strategies non monotonically decreasing the error: binary predicted, highest predicted, popularity, log(pop)∗entropy.

The monotonically error decreasing strategies have overall a better performance (MAE) during the learning process, except at the end. During the iterations 1-40 the best strategy is random, and the second best is lowest predicted. During iterations 40-90 the non monotonic strategies log(pop)∗entropy and popularity are the best performing. Starting from iteration 120 the MAE of popularity, log(pop)*entropy, and of all the prediction-based strategies does not change anymore. This is because these strategies are not able to add any new

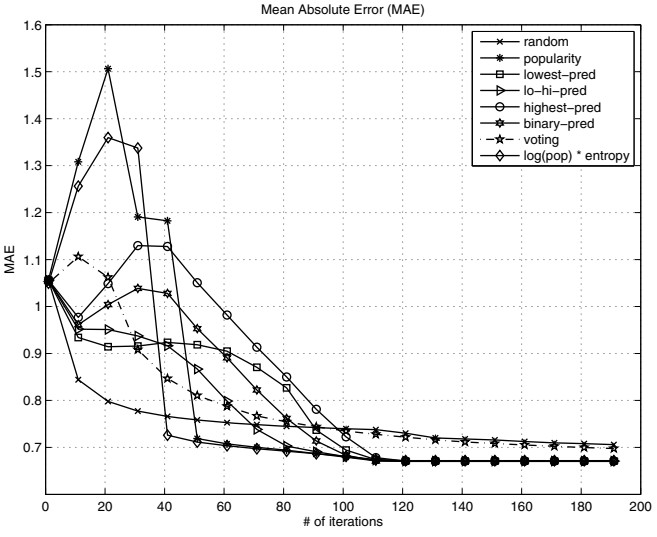

Fig. 1. MAE of the considered strategies (MovieLens data set)

ratings to K. The system MAE during the application of the random and voting strategies keeps decreasing until all the ratings in X are acquired, i.e., moved to K. In fact, it is important to note that prediction based strategies (e.g., highest predicted) cannot elicit ratings for which the prediction can not be made, i.e., for movies and users that don't have ratings in K.

The strategies that are not monotonically decreasing the error can be further divided into two groups. Binary prediction and highest predicted first slightly decrease MAE (iterations 1-10), then they increase MAE (10-30), and finally they keep decreasing the error. While popularity and log(pop)∗entropy, first increase the error (iterations 1-20) and then they keep decreasing it. The explanation for such a behavior is that these strategies have a strong selection bias. For instance, the highest predicted strategy attempts to elicit ratings that have high predicted values. As a result it ends up with adding more high (than low) ratings to the known matrix (K), which biases the rating prediction. This negatively affects MAE at the beginning of the process, because the ratings that they are requesting are more likely to be present in X since it is larger. In a second stage of the process, i.e., after they have collected all the ratings in X with their selection bias, they slowly add the remaining ratings, hence producing in K a distribution of ratings that is closer to the overall distribution in the full data set. In fact, for instance, looking into the data we discovered that at iteration 30 the highest-predicted strategy has already elicited most of the high ratings. Then the next ratings that are elicited are actually ratings with average or low values (but erroneously predicted with high values) and this reduces the bias in K and also the prediction error.

Voting, as explained before, is very much dependent on the strategies that vote for the items. So it can be seen that voting produces an error that is close to the average MAE of the six voting strategies and shows only a minor non monotonic behavior.

4.2 Number of Acquired Ratings

It is important to measure how many ratings are added by the considered strategies. In fact, certain strategies can acquire more ratings, by better guessing what items the user actually experienced. This occurs in our simulation if a strategy asks the simulated user for more ratings present in the matrix X. Conversely, a strategy may not be able to acquire many ratings but those actually acquired are more useful to generate better recommendations.

Table 1 shows the percentage of the requested ratings that have been actually acquired in a particular iteration because present in X. This is a key issue for an elicitation strategy since those strategies that are not able to find movies that users are able to rate will not increase the size of the ratings data set. For instance, random at the iteration 20 can only acquire 3.1% of the requested ratings. Other strategies are more effective, e.g., not surprisingly, binary predicted and highest predicted, at the same iteration can collect more than 20% of the requested ratings.

It is important to note that these ratios underestimates what could be observed in a real scenario. In fact, here X contains all the ratings that can be acquired by a strategy. But, this is only a subset of the ratings that a generic MovieLens user could provide, since it includes only those actually collected by MovieLens. To illustrate better this situation, we conducted a small experiment by extracting a random subset of 50 movies from MovieLens and asking our colleagues (20) to indicate how many movies they could rate. On average they indicated 6 movies, i.e., a ratio of 12%, i.e., more than 4 times larger than the reply ratio of the random strategy in this simulation. This indicates that in reality, users could rate many more movies requested by the various strategies. This is also illustrated by the findings described in [9]. In their live user experiments, popularity strategy (for instance) could acquire on average 50% of the requested

Table 1. The ratio of the ratings acquired over those requested at different iterations

	acquired/requested ratings ratio			
Strategy	iteration=20	iteration=40	iteration=60	iteration=100
Random	3.1%	3.1%	3.1%	2.8%
Popularity	13.8%	11.3	9.9%	7.3%
Lowest predicted	7.7%	8.3%	8.8%	9.5%
Low-high predicted	13.4%	12.6%	12.0%	11.5%
Highest predicted	20.8%	18.7%	16.1%	12.3%
Binary prediction	20.5%	16.9%	15.7%	12.6%
Voting	12.1%	8.3%	6.7%	4.7%
Log(pop)*entropy	13.0%	10.1	10.0%	7.3%

ratings and pop∗entropy, which is similar to our log(pop)∗entropy strategy, could also acquire a similar percentage of ratings. These results clearly illustrate that many of the strategies presented here could already be applicable in a realistic scenario. But obviously there is still space for defining new strategies that can identify a larger percentage of items that users can actually rate.

4.3 Normalized Discounted Cumulative Gain

We measured NDCG on the first top 10 recommendations with not null values in T (of each user) (Figure 2). Random is the best strategy at the beginning of the active learning process, but at iteration 40 voting passes random and then remains the best strategy. Excluding the random strategy, voting and highest predicted are the best overall. Lowest predicted is by far the worst, and this is very different from its performance with respect to MAE. Moreover, another striking difference from the MAE results, is that all the considered strategies improve NDCG monotonically. Analyzing the experiment data we discovered that lowest predicted is not effective for NDCG because it is eliciting more user ratings on the lowest ranked items and this is useless to predict the ranking of the top items. It is also important to note that here voting is by far the best. We should also note that voting and random also have the best behavior in term of coverage (not shown here for lack of space) since they can actually elicit ratings for new items and new users. We have also evaluated the strategies with respect to precision. These results are very similar to those shown previously for NDCG hence due to the lack of space they are not shown here.

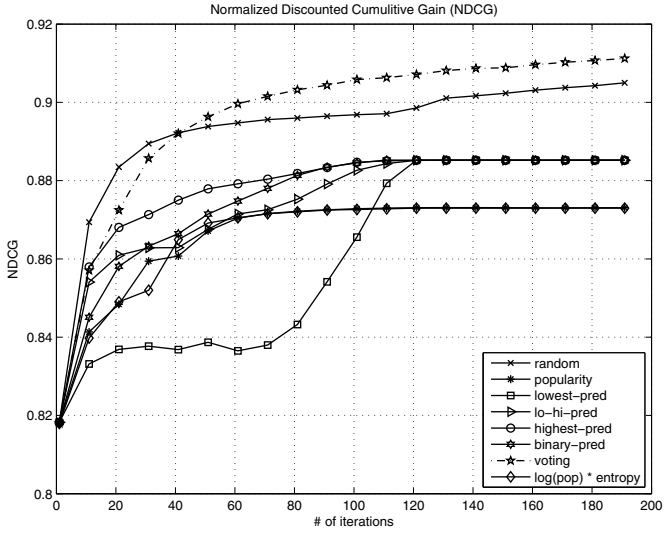

Fig. 2. NDCG of the all strategies

In conclusion from these experiments we can see that *there is not a clear best strategy* that dominates the others for all the evaluation measures (among those that we evaluated). The *voting* strategy is the best for NDCG, whereas for MAE one should suggest *random* at the beginning and successively Popularity and log(pop)*entropy. We observe that voting represents a good compromise which can improve the quality of the ranking of the top items and reduce substantially the prediction error. Similar results have been obtained by running the same experiments on another data set, that is NetFlix: which reinforces the support for the conclusions that we have derived.

5 Related Work

Active learning in RS aims at actively acquiring user preference data to improve the output of the RS [12]. [9] Proposes six techniques that collaborative filtering recommender systems can use to learn about new users in the sign up process. They considered: pure *entropy* where items with the largest entropy are preferred; *random*; *popularity*; $log(popularity) * entropy$ where items that are both popular and have diverse rating values; and finally *item-item personalized*, where the items are proposed randomly until one rating is acquired, then a recommender is used to predict the items that the user is likely to have seen. They studied the behavior of an item-to-item RS only with respect to MAE, and designed an offline experimental study that simulates the sign up process. The process was repeated and averaged for all the test users. In this scenario the $log(popularity) * entropy$ strategy was found to be the best. It is worth noting that these results are not comparable with ours as they measured how a varying set of ratings elicited from one user are useful in predicting the ratings of the same user. In our experiments we simulate the simultaneous acquisition of ratings from all the users, by asking each user in turn for 10 ratings, and repeating this process several times. This simulates the long term usage of a recommender system where users come again and again to get new recommendations and the rating provided by a user is exploited to generate better recommendations to others (system performance).

In [3] is noted that the Bayesian active learning approach introduced in [4] makes an implicit and unrealistic assumption that a user can provide rating for any queried item. Hence, the authors proposed a revised Bayesian selection approach, which does not make such an assumption, and introduces an estimation of the probability that a user has consumed an item in the past and is able to provide a rating. Their results show that personalized Bayesian selection outperforms Bayesian selection and the random strategy with respect to MAE. Their simulation setting is similar to that used in [9], hence for the same reason their results are not directly comparable with ours. There are other important differences between their experiments and ours: their strategies elicit only one rating per request; they compare the proposed approach only with the random strategy; they do not consider the new user problem, since in their simulations all the users have 3 ratings at the beginning of the experiment, whereas in

our experiments, there might be users that have no ratings at all in the initial stage of the experiment; they use a completely different rating prediction algorithm (Bayesian vs. Matrix Factorization).

In [1] again a user-focussed approach is considered. The authors propose a set of techniques to intelligently select ratings when the user is particularly motivated to provide such information. They present a conversational and collaborative interaction model which elicits ratings so that the benefit of doing that is clear to the user, thus increasing the motivation to provide a rating. Item-focused techniques that elicit ratings to improve prediction on a specific item are proposed. Popularity, entropy and their combination are tested, as well as their item focused modifications. Results have shown that item focused strategies are constantly better than unfocused ones. Also in this case, their results are complementary to our findings, since the elicitation process and the evaluation metrics are different.

6 Conclusions and Future Work

In this work we have addressed the problem of selecting ratings to ask users also defined as the ratings elicitation problem. We have proposed and evaluated a set of ratings elicitation strategies. Some of them have been proposed in a previous work [9] (popularity, random, log(pop)∗entropy), and some, which we define as prediction-based strategies, are new: binary-prediction, highest-predicted, lowest-predicted, highest-lowest-predicted. We have also proposed a voting strategy combining six different strategies which shows very good performances for several evaluation metrics (MAE, NDCG, precision, coverage). We have evaluated these strategies for their system-wide effectiveness implementing a simulation loop that models the day-by-day process of rating elicitation and rating database growth. We have taken into account the limited knowledge of the users, i.e., the fact that the users will not know all the possible ratings, and this is a small percentage of all of them. During the simulation we have measured several metrics at different phases of the rating database growth.

The performed evaluation has shown that different strategies can improve different aspects of the recommendation quality and in different stages of the rating database development. Moreover, we have discovered that some strategies may incur the risk of increasing MAE if they keep adding only ratings with a certain value, e.g., the highest-predicted strategy that is an approach often adopted in real RSs. In addition, prediction-based strategies neither address the problem of new users, nor of new items. Whereas, voting, popularity and log(pop)*entropy strategies are able to select items for new users, but can not select items that have no ratings.

In the future we want to study the effect of different prediction algorithms, e.g., K-Nearest Neighbor [11], on the performance of the selected strategies. Moreover, we want to better explore the possibility of combining strategies using different heuristics depending on the state of the target user, and the data set, hence building a more adaptive approach.

References

1. Carenini, G., Smith, J., Poole, D.: Towards more conversational and collaborative recommender systems. In: Proceedings of the 2003 International Conference on Intelligent User Interfaces, Miami, FL, USA, January 12-15, pp. 12–18 (2003)
2. Elahi, M., Ricci, F., Repsys, V.: System-wide effectiveness of active learning in collaborative filtering. In: Bonchi, F., Buntine, W., Gavalda, R., Guo, S. (eds.) Proceedings of the International Workshop on Social Web Mining, Co-located with IJCAI, Barcelona, Spain (July 2011)
3. Harpale, A.S., Yang, Y.: Personalized active learning for collaborative filtering. In: SIGIR 2008: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 91–98. ACM, New York (2008)
4. Jin, R., Si, L.: A bayesian approach toward active learning for collaborative filtering. In: UAI 2004: Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence, Banff, Canada, July 7-11, pp. 278–285 (2004)
5. Koren, Y., Bell, R.: Advances in collaborative filtering. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.) Recommender Systems Handbook, pp. 145–186. Springer, Heidelberg (2011)
6. Liu, N.N., Yang, Q.: Eigenrank: a ranking-oriented approach to collaborative filtering. In: SIGIR 2008: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 83–90. ACM, New York (2008)
7. Manning, C.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
8. McNee, S.M., Lam, S.K., Konstan, J.A., Riedl, J.: Interfaces for eliciting new user preferences in recommender systems. In: Brusilovsky, P., Corbett, A.T., de Rosis, F. (eds.) UM 2003. LNCS, vol. 2702, pp. 178–187. Springer, Heidelberg (2003)
9. Rashid, A.M., Albert, I., Cosley, D., Lam, S.K., Mcnee, S.M., Konstan, J.A., Riedl, J.: Getting to know you: Learning new user preferences in recommender systems. In: IUI 2002: Proceedings of the 2002 International Conference on Intelligent User Interfaces, pp. 127–134. ACM Press, New York (2002)
10. Rashid, A.M., Karypis, G., Riedl, J.: Learning preferences of new users in recommender systems: an information theoretic approach. SIGKDD Explorations 10(2), 90–100 (2008)
11. Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.): Recommender Systems Handbook. Springer, Heidelberg (2011)
12. Rubens, N., Kaplan, D., Sugiyama, M.: Active learning in recommender systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.) Recommender Systems Handbook, pp. 735–767. Springer, Heidelberg (2011)
13. Weimer, M., Karatzoglou, A., Smola, A.: Adaptive collaborative filtering. In: RecSys 2008: Proceedings of the 2008 ACM Conference on Recommender Systems, pp. 275–282. ACM, New York (2008)