Free University of Bozen-Bolzano
Faculty of Computer Science

Master of Science Thesis

User Preferences Elicitation Strategies for Collaborative Filtering

by

Valdemaras Repšys

Supervisor: prof. Francesco Ricci

Bozen-Bolzano, 2009

Contents

Abstract

Recommender systems provide assistance when making a decision about a purchase of a product or a service. They are used by major online retailers to aid customers in finding the most suited products to buy. Such recommender systems during their operation collect information about customer preferences and when requested to provide a recommendation try to find out what the best products for the current purchase are. Customers could be asked to rate the products that they have consumed in the past. Those ratings are then used to build the customer preferences model according to which the products to recommend are chosen by the system. In this kind of systems the accuracy of the recommendation essentially depends on two factors: the effectiveness of the recommendation algorithm and the number of product ratings known by the system. The more product ratings are elicited from the customers, the better and more accurate the recommendation is. Specific techniques, which are defined as rating elicitation strategies, can be used to select the items to be presented to the user for rating. In our work we heuristically defined several rating elicitation strategies and we evaluated them with a particular evaluation methodology that exploits an existing dataset of ratings. We discussed the benefits and drawbacks of different strategies with respect to a range of different aspects of a recommender system (e.g. precision, ranking quality or coverage). We have as well taken into account the stage of the ratings database development. The evaluation showed that different strategies can improve different aspects of the recommendation quality or rating prediction accuracy in different stages of rating database development and that the best strategy depends on the metric one wants to optimize.

Chapter 1

Introduction

1.1 Background

Internet has become indispensable, it provides huge amounts of information on every topic: either news or books, tourism information etc. Every person who wants to find information or perform a transaction on Internet is faced with an overwhelming number of choices. For example, a person who is looking for a movie to watch can choose from more than 1.3 million different titles in the imdb database. Choosing the right product to consume or purchase is an increasingly relevant problem, due to the growing variety and informational globalisation. On the one hand, unlimited choice provides an opportunity for a consumer to acquire the product satisfying his special personal needs, on the other hand the consumer becomes confused when faced with this enormous number of products he can choose from.

Recommender systems [18] aim at solving the problem of unlimited choice [2]. They provide personalized suggestions for digital content, products or services. The task of finding a movie one likes is simplified by the recommender system to the task of choosing among 5 recommendations instead of 1.3 million different titles. Recommender systems help consumers to find the most appropriate products that give more satisfaction than the mainstream products that the consumer would probably choose if there were provided no recommendations.

Recommender systems became an independent research area in the 1990s [1]. Briefly put, the recommendation problem consists of estimating the utility of the items that have not been consumed by the user in the past. Once it is possible to estimate the utility, it is also possible to recommend to the user the items that will potentially be the most useful for him.

Recommender systems can be classified into the following main categories, based on how recommendations are made: content-based [13], collaborative based

[17], demographic [19], utility-based [25], knowledge based [4] and hybrid recommenders [5].

In this thesis we concentrate on collaborative-based filtering techniques.

In collaborative filtering, the predicted ratings are computed only using other ratings given by the users to items and not other knowledge. In other words, collaborative filtering builds a user model using only the users' ratings, and neither the user demographic data nor the description of items that he liked in the past is used. Normally, items with the highest predicted ratings are recommended. Accurate rating prediction has been the most important issue in collaborative filtering recommenders. The two most popular approaches for computing rating predictions are the neighborhood approach [3] and latent factor models [12].

1.2 Problem

The quality of recommendations and the rating prediction accuracy depends not only on the the prediction algorithm, but also on the data available in the rating database. In this work we concentrate on improving and enlarging the set available data.

Collaborative filtering recommender systems use the available item ratings to build the prediction model, which is used to approximate unknown ratings. The more ratings are available and the more information those ratings carry, the higher the prediction and the recommendation accuracy is. Therefore, it is very important for a recommender system to keep acquiring new and useful ratings in order to maintain and even improve the quality of the recommendations.

New ratings can be acquired, for instance, by asking users to rate some items in the registration phase. Normally, users can also navigate through the lists of existing items and rate them if they wish so. There can also be a possibility to rate the items that are recommended to user.

Usually, recommender systems deal with extremely large numbers of items. Consequently, a small set of items has to be carefully chosen to be presented to the user in order to ask him to rate them.

The main problem addressed in this work is the problem of choosing the items to ask users' opinion about. This problem is defined as a rating elicitation problem and a method used to choose the items is defined as rating elicitation strategy.

The problem has been tackled in several previous works [20] [16] [6] [7].

1.3 Research Goals

Rashid et al. [16] introduce several rating elicitation methods (or strategies) for new users and evaluate their performance with regard to the mean absolute error

(MAE) and user effort. The results showed that the best performing strategies are *popularity* and the $log(popularity) * entropy$.

Carenini et al. [6] proposed techniques to elicit ratings to improve prediction on a specific item. Elicitation methods proposed in [16] are extended to become item-focused. Results show that focused strategies are consistently better than unfocused ones with regard to MAE and user effort.

Harpale et al. [7] proposes a personalized form of Bayesian active learning which combines the entropy minimization and a probability of getting a rating. Results show that their personalized Bayesian method works better with regard to MAE than random and the standard Bayesian technique.

McNee et al. [20] compare three interfaces to elicit informatin from new users: having the system choose items for users to rate, asking the users to choose items themselves, and a mixed-initiative interface. By measuring MAE it was found that pure interfaces produce more accurate user models than the mixed model.

The goal of our work is a more systematic study on the application of different rating elicitation strategies. We want to understand the benefits and drawbacks of different strategies with respect to a range of different aspects of a recommender system (e.g. precision, ranking quality or coverage). Moreover, we are interested in understanding if the strategy selection must take into account the size and the state of the rating database. In fact, previous work mostly used MAE to evaluate the rating elicitation strategies, whereas we employ also other metrics, such as precision and normalized discounted cumulative gain (NDCG), which measures how the top recommendations are ranked, and if this ordering respects the user preferences.

We have chosen a set of strategies to evaluate, among which there are some strategies covered in the previous work (*random*, *popularity*, *variance*) and new strategies that were not tested before.

1.4 Approach

In our work we heuristically defined several rating elicitation strategies and we evaluated them with a particular evaluation methodology that exploits an existing dataset of ratings.

Recommender systems research have exploited two ways for evaluating a particular recommendation method: online and offline evaluations. In an online evaluation a real recommender system is run and experiments on real users are performed. Online experiments require to access a recommender system with a large user community. It is rather expensive and time consuming to gain such an access, so most of the experiments are performed offline. Offline experiments exploit already existing rating datasets and do not require access to an online recommender system.

	Starting Phase	Improvement Phase
MAE	Random Binary-prediction-rand	Low-high-predicted Lowest-predicted Random
Precision	Highest-predicted	Binary-prediction Lowest-highest-predicted Highest-predicted
NDCG	Highest-predicted-rand Random	Highest-predicted-rand Random Popularity-rand

Table 1.1: The best performing strategies with respect to the simulation stage and various measures

Therefore, we build our own offline experimental environment. We created software which simulates the process of rating elicitation and rating database growth. To run the simulations we used the freely available MovieLens rating data. In order to compare the performance of different rating elicitation strategies we measured various recommendation quality metrics during the simulation.

1.5 Contribution

In this work we have identified and tested several rating elicitation strategies and compared their performance along a well defined procedure. Together with some strategies evaluated in previous work [16] we propose a set of prediction-based strategies and strategy combinations, which are defined partially randomized strategies. Prediction-based strategies select items to elicit based on their predicted rating. The highest-predicted strategy, for instance, selects items with the highest predicted ratings, lowest-predicted - with the lowest predicted ratings. Partially randomized strategies combine the random strategy, when the items to elicit are selected randomly and which has been proved to be effective in expanding the competence of the recommender system, with any other strategy. For instance, partially randomized highest-predicted selects items with the highest predicted ratings and also includes some random items. It also selects random items when the ratings can not be predicted for new users.

We show that different strategies improve different aspects of the recommendation quality or rating prediction accuracy in different stages of rating database development. We used a set of different evaluation measures to capture different aspects of improvement: MAE to measure the improvements in prediction accuracy, precision and normalized discounted cumulative gain (NDCG) to measure the relevance of recommendations and the quality of produced ranking.

Table 1.1 lists the best performing strategies with respect to the chosen measures and the phase of the evaluation. In the starting phase there are many new users and few items. The improvement phase reflects more mature state of the ratings database where, if a strategy managed to address a new user and new item problems, there are very little or no users and items with no ratings.

To sum up, the main contributions of our work to this research field are:

- A large set of rating elicitation strategies.
- The evaluation of the strategies behavior with respect to a wide set of metrics (MAE, Precision, NDCG, coverage).
- Some guidelines for the selection of the strategy depending on the rating database size and target evaluation metric.

1.6 Outline of the Thesis

Firstly, we introduce the state of the art of the recommender systems and the main rating prediction methods (chapter 2) and describe the problematics of rating elicitation and the goal of our work (chapter 3). Then we overview the main work that has been done in the area of learning user preferences (chapter 4) and define the rating elicitation strategies we chose for testing (chapter 5). Finally, we describe the experimental setup we have built to conduct the experiments on elicitation strategies (chapter 6) and present and discuss the experimental results (chapter 7). Conclusions and future research directions are provided in chapter 8.

Chapter 2

State of the Art

2.1 Recommender systems

Recommender systems are classified by Burke [5] into the following main categories, based on how recommendations are made:

- Content-based recommendations: in a content-based system, the objects of interest are defined by their associated features. For example, text recommendation systems use the words of their texts as features. The user is recommended items similar to the ones the user preferred in the past.
- Collaborative-based recommendations: items with highest predicted ratings are recommended. The predicted ratings are computed only using other ratings given by the users to items and not other knowledge.
- Demographic recommendations: this method generates recommendations by identifying demographically similar users and using their ratings. Those methods aim to categorize the user based on personal attributes and make recommendations based on demographic classes.
- Utility-based: recommendations are based on an evaluation of the match between a users need and the set of options available. Utility-based recommenders make suggestions based on a computation of the utility of each object for the user.
- Knowledge-based: attempts to suggest objects based on inferences about a users needs and preferences. Knowledge-based approaches are distinguished in that they have functional knowledge: they have knowledge about how a particular item meets a particular user need, and can therefore reason about the relationship between a need and a possible recommendation.

- Hybrid recommendations: these methods combine collaborative-based and content-based approaches.

In this thesis we concentrate on collaborative-based filtering techniques. The first collaborative filtering systems that used a database of historical user ratings to automatically match each individual to others with similar opinions were GroupLens [17] [11], Ringo [21] and Bell-core's Video Recommender [9]. Grouplens recommender system worked in the domain of newsgroup articles, Ringo operated in the domain of music and musical artists.

The task in collaborative filtering is to predict the usefulness or relevance of items for a particular user. In collaborative filtering-based approach recommendations are derived from the ratings of other users and the ratings provided by the target user.

There are two classes of collaborative filtering algorithms [3]: memory-based and model based. Memory-based algorithms employ directly the rating database to compute the rating predictions. Model-based algorithms, in contrast, use available ratings build build an offline prediction model, which is then used to compute predictions online.

The input for the collaborative filtering algorithms is a set of ratings $R = \{r_{u,i}\}$ provided by users $U = \{u_1..u_N\}$ for items $I = \{i_1..i_M\}$.

In collaborative filtering, the two most popular approaches for computing rating predictions are: the neighbourhood approach and latent factor models.

2.2 Neighbourhood Approach

To predict an opinion of the target user for the item two sets of data are needed: previous opinions of the user for other items and the opinions of other like-minded users. In the neighborhood approach the item's rating predictions are based on how the item was rated by the users similar to the target user. Then the rating $r_{u,i}$ for the user u and the item i is predicted in the following way:

$$r_{u,i} = \bar{r}_u + \frac{\sum_{u' \in U'} s(u,u')(r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in U'} \mid s(u,u') \mid} \tag{2.1}$$

where $\bar{r}_u$ denotes the average of user's u ratings, $s(u,u')$ is a similarity measure between two users and U' is a set of users similar to user u. The similarity between users is computed using the Pearson correlation [17]

$$s(u,u') = \frac{\sum_{i \in I_{u,u'}} (r_{u,i} - \bar{r}_u)(r_{u',i} - \bar{r}_{u'})}{\sqrt{\sum_{i \in I_{u,u'}} (r_{u,i} - \bar{r}_u)^2 \sum_{i \in I_{u,u'}} (r_{u',i} - \bar{r}_{u'})^2}} \tag{2.2}$$

where $I_{u,u'}$ denotes the set of items both users u and u' have rated.

2.3 Latent Factor Models

Latent factor models offer a different approach to rating prediction. One of the examples of latent factor models is singular value decomposition (SVD) model [12]. SVD assigns both users and items with factor vectors of the same size. Those vectors are automatically inferred from the user feedback. Each element of the factor vector assigned to an item reflects how well the item represents a particular latent aspect. For example, when the products are movies, the factors might measure such aspects as comedy, drama, action or even other uninterpretable dimensions. User factor vectors measure the preference of the user for each factor.

The task of the factorization is to split the matrix of ratings R into two other matrices S and M in such a way that:

$$R \approx SM^T \tag{2.3}$$

Where S is an $\mid U \mid \times F$ and M is an $\mid I \mid \times F$ matrix. F represents the number of factors we wish to use. The value s_{uf} denotes how much the user u likes the factor f and the value m_{if} denotes how strong the factor f is in the item i.

In the experiments of the thesis project an implementation of regularized SVD technique [22] was employed for predicting ratings. Regularized SVD is a technique proposed for collaborative filtering by Simon Funk [23]. In the regularized SVD the predictions for the ratings are computed in the following way:

$$\hat{r}_{ui} = \sum_{f=1..F} s_{uf} m_{if} \tag{2.4}$$

The factor matrices are esimated by minimizing the sum of error using regularization and early stopping.

$$e_{ui} = r_{ui} - \hat{r}_{ui} \tag{2.5}$$

$$s_{uf} = s_{uf} + lrate * (e_{ui} * m_{if} - \lambda s_{uf}) \tag{2.6}$$

$$m_{if} = m_{if} + lrate * (e_{ui} * s_{uf} - \lambda m_{if}) \tag{2.7}$$

After features are learned the predictions are trimmed to be in the range from 1 to 5. We use the following parameters in our experiments: the learning rate $lrate = 0.001$,regularization parameter $\lambda = 0.015$ and the number of features $F = 96$.

Chapter 3

Problem Definition

Recommendation quality is an essential aspect of every recommender system. Useful recommendations improve customer satisfaction and loyalty resulting in increased sales. The quality of recommendations and the rating prediction accuracy depend on two key factors: the the prediction algorithm, the data available in the rating database. The better algorithm is used, the more accurately ratings can be predicted and the more suitable items can be recommended to users. However, in this work we concentrate on improving and enlarging the set available data.

Collaborative filtering recommender systems use available item ratings to build the preference prediction model. The model is used to approximate the ratings for items that have not yet been rated. The more ratings there are and the more information those ratings carry, the higher the prediction and the recommendation accuracy is. Therefore, it is essential for a recommender system to keep acquiring new and useful ratings in order to maintain and even improve the quality of recommendations.

Usually, rating databases are dynamic entities and grow over time. As an example, assume a movie recommender system: new movies as they are released and new users as they join the system are continually added to the database, as well as already existing users express their opinions about movies and add more ratings to the database. The more ratings are known to the system, the more accurate predictions the algorithm can make. The process of improvement of prediction accuracy and other measures as the number of known ratings grows is depicted in figures 3.1, 3.2, 3.3, 3.4. For creating the figures the dataset (which is described in chapter 6.1 was split into three parts: training data (1000 ratings), unknown data (69,000 ratings) and testing data (30,000 ratings). Then for every iteration 1000 randomly chosen ratings from the unknown data were added to the training data and measurements (described in chapter 6.3) were recorded afterwards.

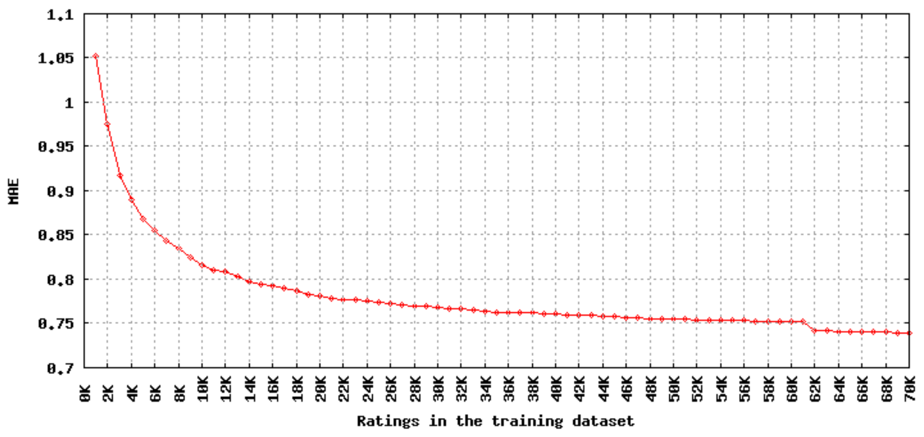

Figure 3.1: Change of MAE when adding more ratings to the training dataset

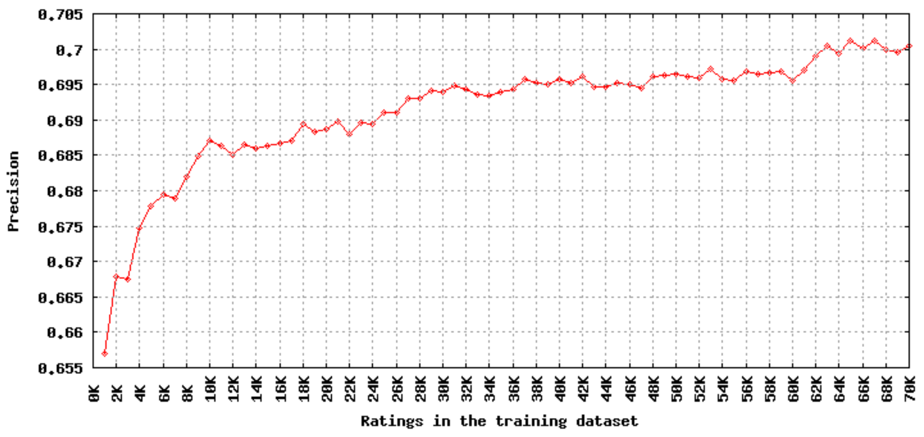

Figure 3.2: Change of precision when adding more ratings to the training dataset

However, the prediction accuracy depends not only on the number of available ratings, but also on their nature. Ratings for some items may be more useful than ratings for other items. For instance, in a movie recommender system, the knowledge that a user has given a high rating to "The Lord of the Rings" might not be very useful. The reason for this is that almost all other users have rated the same movie high and thus the rating does not carry much information that would allow to distinguish the target user from other users or reveal his personal tastes [6].

Recommender systems usually present items to users for for two purposes:

1. to recommend items a user might decide to buy or be interested in.

2. to ask a user to rate items.

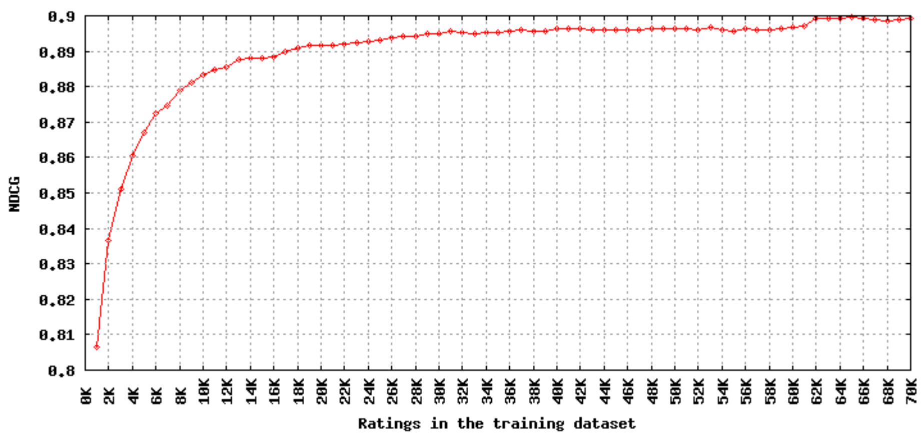

Figure 3.3: Change of normalized discounted cummulative gain when adding more ratings to the training dataset

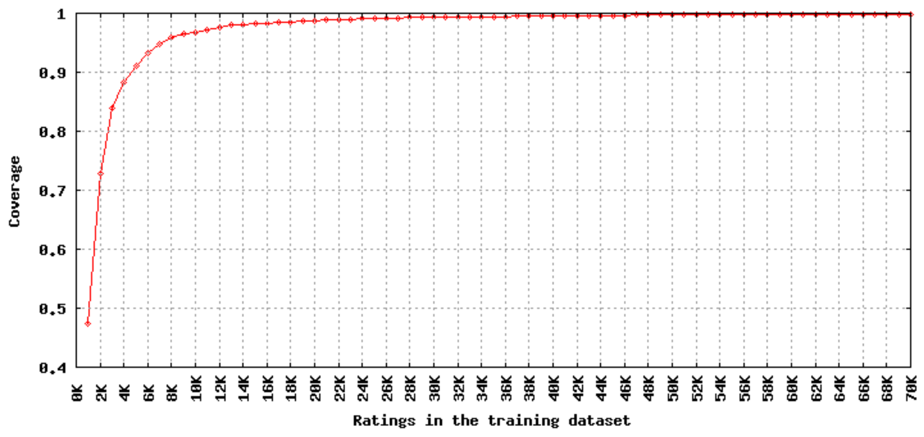

Figure 3.4: Change of coverage when adding more ratings to the training dataset

Most often users are asked to rate items during the registration phase in order to learn about their preferences and build the user model. Normally, users can also navigate the item catalogs and rate items if they wish so.

There can also be a possibility to rate items that are presented to user during the recommendation process, meaning that a user can rate items that were recommended to him if he already consumed them or is otherwise familiar with some of those items. In this case the the recommendation process is both explorative and exploitative. It explores the user tastes to build more accurate prediction model, and exploits by recommending items that a user is likely to buy. It is up to the system to decide on which rating acquisition methods to use.

Usually, recommender systems deal with extremely large numbers of items, for this reason a small set of items has to be carefully chosen to be presented to the user in order to ask him to rate them.

Recommender systems can employ different strategies for choosing items to be presented to user to ask for his opinion about it. The most basic method is randomly asking users to rate items. In fact, random is quite a good method for acquiring ratings, since it does not introduce any bias into the database. Other methods can be also used: for instance, *MovieLens Classique* strategy, discussed in [16], presents new users with pages of items to rate, where each page contains one item selected randomly from manual list of popular items and the rest is selected randomly from the set of all items. This way it tries to increase the chance that users can rate at least one movie per page.

Presumably, applying different rating acquisition strategies results in various database rating growth patterns and different growth rates. Distinct growth patterns should create different effects on various recommendation quality and prediction accuracy measures.

For instance, a specific strategy might work very well in increasing the coverage i.e. introducing new items and eliciting ratings for new users, but perform not so well in increasing the prediction accuracy (MAE). Another strategy might be very good in improving the recommendation quality (precision and NDCG), but perform worse with regard to prediction accuracy. Distinct strategies might also affect the measures in a different way depending on the size of existing training dataset.

Rating acquisition strategies should be chosen in accord with the goals of the recommender system to influence the growth of the database in the most beneficial way for a specific recommender system. Naturally, in businesses, the main goal of the recommender system is the user satisfaction and sales increase, what depends a lot on the usefulness of the recommendations. However, there might be different types of usefulness. For instance, it could be more beneficial for a certain recommender system to increase the prediction accuracy for all items in the database, for another system it could be more beneficial to increase the prediction accuracy only for top 3 items and a certain error is allowed in the low rated item predictions. Other recommender systems might need to maximize the prediction accuracy for top 10 items, where the rank of the item, if it falls within top 10, is of no importance.

The higher is the growth rate of the training dataset, the greater should be the effect of the particular rating acquisition strategy. However, growth patterns should mostly depend on the kind of ratings that are acquired and added to the rating database.

Chapter 4

Related Work

Rating elicitation problem can be considered as an active learning problem. Active learning aims at actively acquiring training data to improve the ouput of the recommender system. Most of the research that has been done in this area concentrates on the problem of new users.

Rashid et al. [16] proposes six techiques that collaborative filtering recommender systems can use to learn about new users. Those techniques select a sequence of items to present to each new user for rating. Those techniques are: pure entropy (items with highest entropy are selected for elicitation), random (items are selected randomly), popularity (items with most ratings are selected), $log(popularity) * entropy$ (items are selected for elicitation according to score which is computed by multiplying log(popularity) and entropy values) and item-item personalized (items are proposed randomly until one rating is acquired, then a recommender is used to compute items that the user is likely to have seen).

To measure MAE an offline experimental design which simulated the sign up process was used. Each strategy was used to select a number of items to each user without knowing if they are available for that user in the dataset or not. Then MAE was measured for each user using elicited ratings as the training data and all other ratings as testing data. It was shown that $log(popularity) * entropy$ method was the best with regard to improving MAE values. However, popularity and item-item personalized strategies outperformed $log(popularity) * entropy$ with regard to the user effort. User effort was computed as a fraction of items elicited over total number of items presented.

Carenini et al. [6] propose a set of techniques to intelligently select what information to elicit from the user in situations where the user may be particularly motivated to provide such information. They present a conversational and collaborative interaction model which elicits ratings in such a way as to make the benefit of doing so 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 is tested, as well as their item focused modifications. The item focused techniques are different from the classical ones in that popularity and entropy are not computed on the whole rating matrix, but only on the matrix of user's neighbors that have rated an item for which the prediction accuracy is aimed at being improved. Results have shown that item focused strategies are constantly better than unfocused ones.

Harpale et al. [7] point out that Bayesian active learning approach in collaborative filtering makes an implicit and unrealistic assumption that a user can provide rating for any queried item. They propose a new approach which does not make such an assumption. An extension to Bayesian selection approach is presented which introduces a probability that a user has consumed an item in the past and is able to provide a rating. They evaluated the active learning techniques in an offline simulation which imitates the rating elicitation process and measures the changes of MAE on every iteration. Their results showed that the personalized Bayesian selection outperformed Bayesian selection with regard to MAE.

McNee et al. [20] address a more general problem. They try to answer a question of what is better: allowing a user to enter the items and their ratings by themselves, proposing a user with a list of items and asking him to rate them or using a method which combines the two approaches. In other works, they compare three interfaces to elicit information from new users: having the system choose items for users to rate, asking the users to choose items by themselves, and a mixed-initiative interface. They performed an online experiment, which showed that that pure interfaces produce more accurate user models than the mixed model with regard to MAE.

In our work, together with some strategies also evaluated in [16] we propose a set of new prediction-based strategies and strategy combinations, which we define as partially randomized strategies. The simulation process used in our work is similar to one used in [7], because it makes many iterations of presenting users with items to rate and measures the changes of MAE on every iteration. One of the differences is that they assume that all users have 3 ratings in 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.

The approach in [6] is specifically designed for classical neighborhood collaborative filtering techniques, whereas in our work we use matrix factorization techniques. Moreover, in our work we used a set of metrics to measure the strategy performance, which include MAE, precision, NDCG and coverage, whereas, previous work mostly used MAE. However, we do not discuss the user effort, which was done in some previous work [16] [6].

Chapter 5

Rating Elicitation Strategies

A rating elicitation strategy S represents a particular order in which the ratings for items are requested to the user. S can be seen as a function $S(u, N, K, Unclear)$ which returns a list of items to be elicited (L), where u is a user for which the elicitation list is to be generated, N is the number of items to be elicited (the size of L), K is the known ratings for all users, $Unclear$ is the set of items for which the ratings are not yet known by the user u and it is not known whether they are available or not. As it will be shown in the next chapter, $Unclear = (All \setminus Items_u) \setminus Asked_u$, where All represents the set of all items, $Items_u$ represents all the items such that their rating of the current user u is available in K and $Asked_u$ contains the items that have already been chosen for elicitation in the past for the current user. The $Unclear$ set is used to avoid asking user to rate: ratings that are already known and ratings that are known to be not known by the user.

Every strategy analyzes the dataset of already known preferences and scores each item from the $Unclear$ dataset. N items with highest score are returned for acquisition.

Commonly, users can provide ratings for random items presented to them during the registration phase or while browsing the item catalogs at a later phase. Usually there is a possibility to rate items that are recommended as well. Therefore, we consider random and highest predicted to be the traditional approaches to rating elicitation.

5.1 Individual Strategies

We present the following individual strategies:

- Popularity: the score for an item is the number of times it appears in the

K dataset.

- Binary Prediction: every preference in the K dataset is assigned a rating value 1. For each user-item pair when the preference for that pair does not exist in K a preference with 0 rating value is inserted creating a new preference set B. An SVD model is built using B as training data and ratings are then predicted for each item in the $Unclear$ dataset for user u. The score for an item is the predicted value for that item.

- Highest Predicted: ratings are predicted for all items in the $Unclear$ and the score is the predicted value.

- Lowest Predicted: ratings are predicted for all items in the $Unclear$ and the score $= MX - predictedvalue$, where MX is maximum possible rating (In the case of Movie Lens dataset it is 5).

- Highest and Lowest Predicted: ratings are predicted for all items in the $Unclear$ set. A score is $|\frac{MX-MN}{2} + MN - predictedvalue|$, where MX is maximum possible rating and MN is minimum possible rating (In Movie Lens dataset 1 and 5, respectively).

- Random: the score for an item is a random integer number from 1 to 10

- Variance: the score is equal to the item ratings' variance in the K dataset.

5.2 Partially Randomized Strategies

We present strategies a particular kind of strategies, which we define as partially randomized strategies. Partially randomized strategies combine the random strategy, when the items to elicit are selected randomly, with any other individual strategy. Random extensions R randomizes the result set L of strategy S. We define R as $R(S(u, N, K, Unclear), P)$ which returns the result set L_e, and where P is a randomization level. Random extension R extend the strategy S in the following way:

1. $L = S(u, N, K, Unclear)$ is obtained

2. if L is an empty set, meaning that the strategy S for some reason could not generate the elicitation list, $L_e = \{i_1, .., i_N\}$ where i_n is a random item from the $Unclear$ set. This can happen, for instance, if S is a highest predicted strategy, and predictions can not be obtained for user u.

3. if $|L| < N$, $L_e = L \cup \{i_1, .., i_{N-|L|}\}$, where i_n is a random item from the $Unclear$ set.

4. if $|L| = N$, $L_e = \{l_1, .., l_M, i_{M+1}, ..i_N\}$, where l_n is an n-th element of L, i_n is a random item from the *Unclear* set and M is calculated in such a way that P is a percentage of random items in L_e.

Chapter 6

Experimental Setup

6.1 Testing Data

For performing tests Movie Lens [15] rating database was employed. The dataset consists of 100,000 ratings from 943 users and 1682 movies. Values representing the rating insertion time fall within the interval from 20/09/1997 05:05 to 23/04/1998 01:10.

In order to set up the experiments the testing data had to be explored. The figures below reveal the time related properties of the data. Figure 6.1 depicts how rating insertion to the database is spread over time. Time span between the insertion of the first and the last ratings is split into intervals of 30 days and represented by the x axis in the figures above. Figure 6.2 demonstrates the distribution of new users' appearance over time. We define the user's appearance time as the time when the first user's rating was inserted. Analogically, the figure 6.3 depicts the appearance of new items in the database.

According to figure 6.3 most of the new items fall into the first time interval, while few of them appear in the database at latter intervals. This distribution reflects the evolution of items set in time. As time passes there are more and more new products available.

During data analysis Movie Lens usage pattern was observed: most of the users enter many ratings shortly after their registration and and enter no ratings later. The pattern explains why the distribution of items added (figure 6.1) and the distribution of new users (figure 6.2) correlate. The more new users register to the system the more new ratings are added at that interval of time.

Therefore, we conclude that observed pattern of new users' appearance does not reflect the usual evolution of rating databases. Commonly, the users should add some ratings in the registration phase, but should also keep using the system further and add more ratings from time to time.

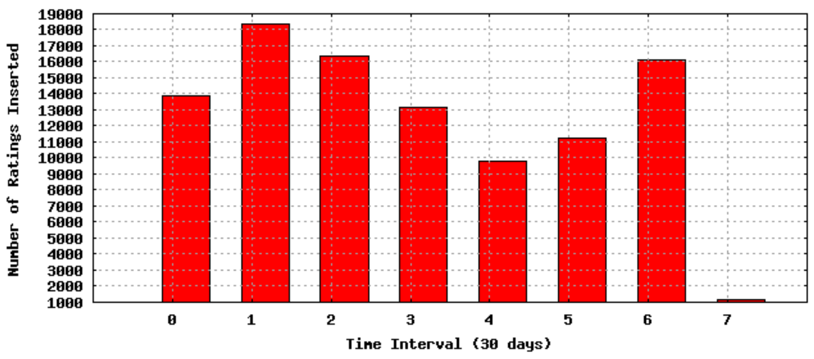

Figure 6.1: Ratings Inserted over Time

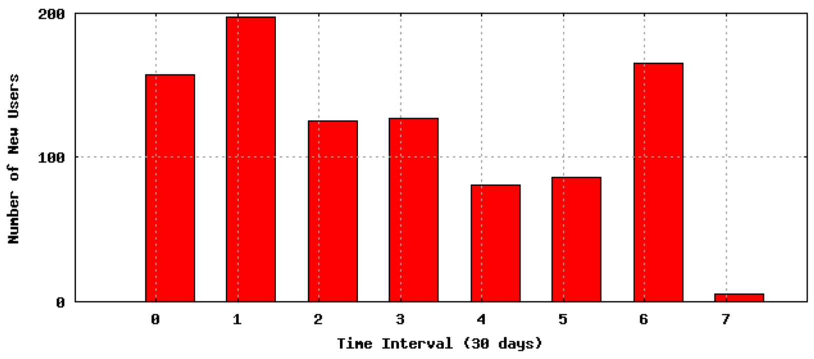

Figure 6.2: Appearance of New Users over Time

Therefore, the random split of the dataset was preferred over of the split according to the rating insertion time. Random split divides the dataset in such a way that the ratings of each user are uniformly distributed over the different parts of the split. Whereas the splitting over time would result in ratings of different users being mostly in one part of the split. This would hinder obtaining useful experimental results, because, for instance, most of the users in the testing dataset would not have at all or would have very little ratings in the training dataset i.e. training and testing datasets would contain different users.

The distribution of ratings of the Movie Lens dataset (figure 6.4) reveals that users tend to provide more ratings for movies that they liked than the movies they didn't like. Obviously, users have a tendency to watch movies that they think they will like and avoid movies that they think they will not like. Therefore, it is reasonable to believe that this is the reason why there are many more high ratings than low ratings.

It will be important to know the distribution of ratings for understanding and interpreting the results of the experiments. Since we split the data randomly for most of our tests all the parts of the dataset should obtain the same distribution

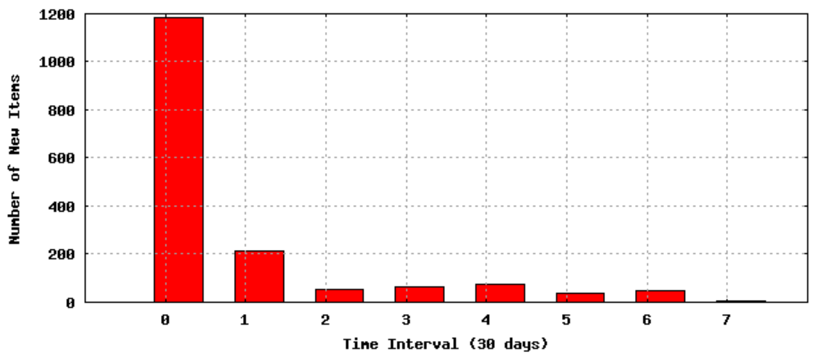

Figure 6.3: Appearance of New Items over Time

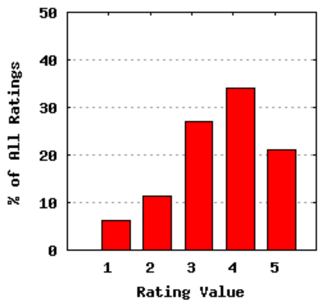

Figure 6.4: The distribution of ratings

as the whole dataset. Therefore, for instance, the ratings distribution of the testing dataset gives us insight that ratings with values 4, 3 and 5 have more weight than ratings with values 1 and 2 in overall MAE calculation.

6.2 Data Splitting Approaches

For each tests the data was split into three different parts:

- K: ratings that are known by the system initially.
- X: ratings that are assumed to be known by the users, but not known by the system initially.
- T: ratings used as a testing dataset.

We use the following two schemes for splitting the data for our experiments:

- *random*: data is split completely randomly. Splitting the data completely randomly creates some users and some items that don't have ratings initially in the known dataset K. This way it provides us with more realistic experimental setup by addressing the problem of new users and new items.

- *random-all*: data is split randomly in such a way that each user and each item has at least one rating in the K dataset. We use this split to run some experiments for seeing the performance of the strategies when they don't have to address the problem of new users and new items.

6.3 Evaluation Measures

To evaluate the rating acquisition strategies measures had to be defined. We chose three measures: mean absolute error (MAE), precision and normalized discounted cumulative gain (NDCG) and coverage.

6.3.1 Mean Absolute Error

MAE is the most common measure for evaluating rating prediction algorithms. It measures the average absolute deviation between a predicted rating and the users true rating [8] or, in other words, how close rating predictions are to the real values of ratings. In this work we use it to evaluate rating elicitation strategies.

MAE is computed using the following formula:

$$MAE = \frac{1}{|T|} \sum_{r_{u,i} \in T} |r_{u,i} - \hat{r}_{u,i}| \tag{6.1}$$

where $r_{u,i}$ stands for a real rating value and $\hat{r}_{u,i}$ represents the predicted value for user u and item i. The MAE equation sums the absolute values of prediction error for all ratings in the testing data set T.

6.3.2 Precision

Each user is assigned two sets of items L_u and $\hat{L}_u$. Where L_u is a is a set of items from testing dataset T that have been assigned a rating value 4 or 5 by the user u (relevant items), and $\hat{L}_u$ is a hit set, containing 10 items from T with highest predicted rating values for the user u.

Precision P is computed using the following formulas:

$$P_u = \frac{|L_u \cap \hat{L}_u|}{|\hat{L}_u|} \tag{6.2}$$

$$P = \frac{\sum_{u \in U} P_u}{|U|} \tag{6.3}$$

6.3.3 Normalized Discounted Cumulative Gain

Discounted cumulative gain (DCG) is a measure originally used to evaluate effectiveness of a Web search engine algorithms or other information retrieval applications [10]. In information retrieval discounted cumulative gain measures the relevance of documents in the result set. It employs graded relevance scale to measure the usefulness based on the position of the document in the result list.

It has also been used in the field of collaborative filtering [24] [14]. In collaborative filtering the relevance is measured by the rating value of the item in the predicted recommendation list. The higher is the value of the rating, the better the recommendation is. DCG gives more importance the items in the top of the recommendation list and less to the items at the bottom of the list.

DCG is defined as:

$$DCG_u = r_u^1 + \sum_{i=2}^{p} \frac{r_u^i}{log_2 i} \tag{6.4}$$

where r_u^i is a rating ranked i for user u and p is the length of the recommendation list.

There is an alternative approach for calculating DCG which places much stronger emphasis on having high rated items at the top of the recommendation list:

$$DCG_u = \sum_{i=1}^{p} \frac{2^{r_u^i}}{log_2(i+1)} \tag{6.5}$$

However, we preffered the first approach for computing DCG_u because of it does not put so much emphasis on having high ratings at the top.

The ratings of each user vary in distribution, so the DCG values for different users are not directly comparable. Therefore, cumulative gain for each user should be normalized. This is done by computing ideal $IDCG_u$ for user u. $IDCG_u$ stands for the list of items of size p which contains the highest user's u ratings in the testing dataset. Normalized discounted cumulative gain for user u is then calculated in the following way:

$$NDCG_u = \frac{DCG_u}{IDCG_u} \tag{6.6}$$

In the experiments we measure the average discounted cummulative:

$$NDCG = \frac{1}{|U|} \sum_{u \in U} NDCG_u \tag{6.7}$$

where $|U|$ is the set of users for which the $NDCG_u$ was computed.

In practice, it is not always possible to obtain the predicted recommendation list or the ideal recommendation list of the desired size p. For instance, if for a specific user there are n ratings in the testing dataset and $n < p$, the $IDCG$ of size p can not be obtained. In this case we create an $IDCG_u$ of size n and make $p = n$ for that user. Since the testing dataset does not change we ensure that $IDCG_u$ is the same during a single experiment.

In the cases when either of the recommendation lists can not be obtained, we skip the user and exclude him from the calculation of the $NDCG$. This might occur when there are no ratings for that user in the testing dataset or when predictions can not be obtained, because there are no ratings in the training dataset.

6.3.4 Coverage

The coverage of a recommender system is a measure of the proportion of items in the system over which the system can form predictions or make recommendations [8].

For instance, usually, a recommender system can not predict ratings for new users and new items. Recommender systems that have lower coverage might be limited in the number of items that can be recommended.

We measure coverage using the following formula:

$$coverage = \frac{|\hat{T}|}{|T|} \tag{6.8}$$

where $|T|$ is the size of the testing dataset and $|\hat{T}|$ is the size of the predicted ratings set. In other words, $|\hat{T}|$ represents the number of ratings in the testing dataset T for which we can predict a value.

6.4 Experimental Procedure

Unclear set represents all preferences that are not known by the system, but might be known by the user. The following tests not only reflect how good the strategies are for finding the most useful items, but they also show how effective they are in finding items for which the ratings are available.

The experiment of the strategy S proceeds in the following way:

1. Preferences are split into the known by the system (K), known by the user but unknown by the system (X) and testing data (T).

2. Initial MAE, Precision and NDCG are measured.

3. For each user u:

   a) *All* represents the set of all items.

      $Items_u$ represents all the items such that their rating of the current user u is available in K.

      $Asked_u$ contains the items that have already been chosen for elicitation in the past for the current user u.

   b) Unclear items $Unclear = (All \setminus Items_u) \setminus Asked_u$. Unclear items set contains all items for which there are no ratings available in the known dataset K for the current user and that have not been chosen for elicitation in the past.

c) Using strategy S a group of preferences L is acquired from the *Unclear* items set.

   $L = S(Unclear)$

d) Items chosen for elicitation are added to the asked set.

   $Asked_u = Asked_u \cup L$.

e) L_e, which contains only preferences of items from L that have a rating in X_u is created. X_u is a set of user's u preferences from X set.

   $L_e = L \cap X_u$

f) Add L_e to the set of all elicited preferences L_{all}.

   $L_{all} = L_{all} \cup L_e$

4. Add all elicited preferences to the known preferences set K and empty the elicited preferences set.

   $K = K \cup L_{all}$, $L_{all} = \emptyset$

5. Measure MAE, precision and NDCG and the number of ratings added $\sum_{u \in U} |L_e|$ where U is the set of all users.

6. Repeat steps 3-5 for I times.

Chapter 7

Experimental Evaluation

In this chapter experimental results are presented. The results were obtained measuring MAE, precision, NDCG, the number of ratings added and coverage on each iteration of the test described in chapter 6.4.

The main results of experiments that used both, random and random-all split of data are presented in this chapter. It is important to note that when data is split randomly, some items, as well as some users, may appear only in one part of the split. The experiments were conducted using the following parameters:

- random split: $|K| = 2000$, $|X| = 68,000$, $|T| = 30,000$, $|L| = 5$
- random-all split: $|K| = 2280$, $|X| = 67,720$, $|T| = 30,000$, $|L| = 5$

The sizes of random split datasets and the sizes for random-all split are averages. Splitting the data using random-all results in slightly different K and X dataset sizes every time. Regardless the split used number of iterations was $I = 50$, number of factors: 16. Experiments that used random split were performed 10 times, and ones that used random-all split were performed 4 times. Results presented in this chapter are averages of those tests.

7.1 Individual Strategies

7.1.1 MAE

The performance of individual strategies in this chapter is discussed. The value of MAE during the experimental process is depicted in figure 7.1. According to the change of MAE during the experiment the performance of the strategies can be divided in two groups:

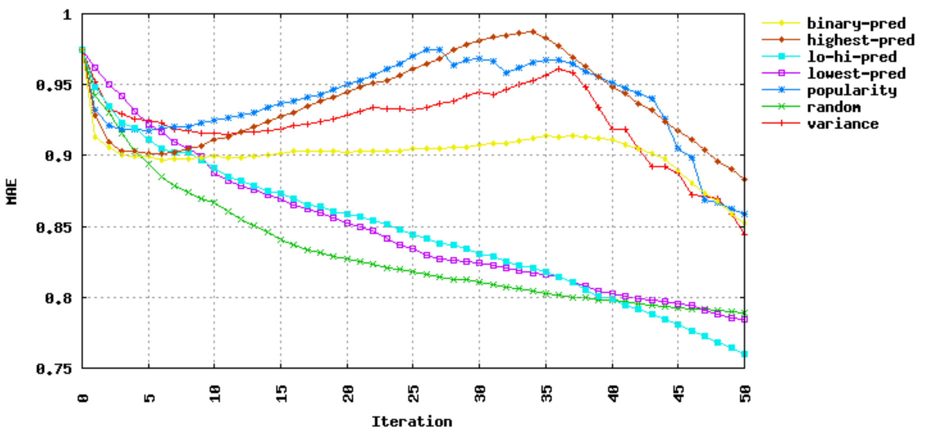

Figure 7.1: MAE; individual strategies; random split

1. Monotone: strategies that show a gradual decrease in MAE value: lowest-highest predicted, lowest predicted and random.

2. Non-monotone: Strategies that have different stages of rising and falling: binary predicted, highest predicted, popularity, variance.

Strategies of the first group show better performance with regard to MAE for all the duration of the test except at the beginning. During the iterations 1-4 the best performing strategy is binary-prediced, the second best being highest predicted, both belonging to the second group. During iterations 5-40 random strategy has the lowest MAE value and it is overtaken by the lowest-highest-predicted strategy at iteration 41.

The performance of the strategies belonging to the second group can be divided into three stages: MAE decreasing in the beginning (approximately iterations 1-5), slowly increasing and reaching the peek in the middle phase (approximately iterations 6-35) and slowly decreasing till the end of the experiment (approximately iterations 36-50).

Obviously, the strategies belonging to the second group have a bias which negatively affects MAE at the second stage of the experiment. The typical behavior of the strategies in the second group is the most strongly shown in highest-predicted strategy, so we analyze its details in order to understand the reason for the observed behavior.

The highest predicted strategy attempts to elicit items that have high predicted rating values. As a result when the system keeps using this strategy it ends up with more high (than low) ratings in the known dataset (K). Low rated movies are selected for elicitation by the highest predicted strategy in cases where prediction algorithm fails to predict a rating value correctly, specifically, when a high rating is predicted for a low rated item. The actual content of the known

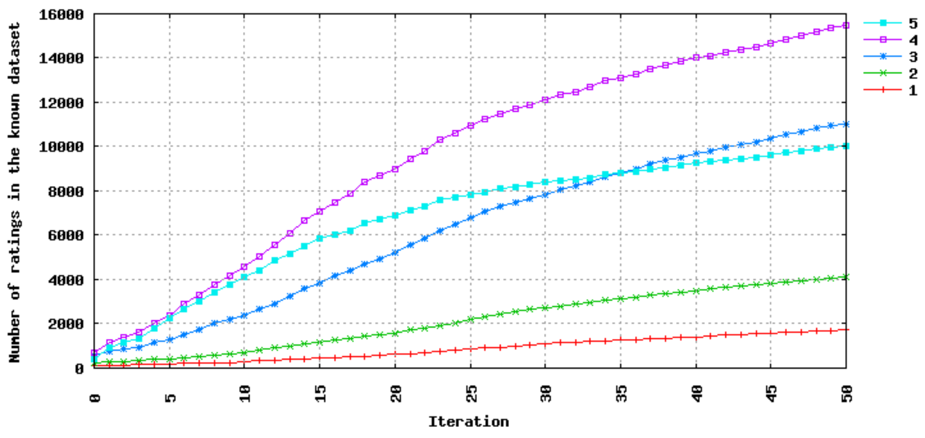

Figure 7.2: Number of different ratings in K dataset depending on the iteration on the process of the highest predicted strategy experiment

dataset (K) during the process of the highest predicted strategy experiment is depicted in figure 7.2.

One can clearly see that in the iterations 0-35 there are many more ratings with value 5 added to the dataset than ratings with value 3 ratings. In fact, in the beginning the number of fives added is almost equal to the number of fours, which yields the deviation of the ratings distribution in the K set from the distribution of the whole Movie Lens dataset and from the distribution of the testing dataset T.

As has been shown in figure 6.4 the distribution of ratings in the Movie Lens dataset is the following:

- 1: 6110 ratings, 6%
- 2: 11370 ratings, 11%
- 3: 27145 ratings, 27%
- 4: 34174 ratings, 34%
- 5: 21201 ratings, 21%

As the experiment proceeds the iteration 35 there are more ratings in the set K having value 3 than those having value 5 and the distribution of ratings gets more similar to the real distribution of the whole dataset and the MAE value starts decreasing again.

In order to understand how adding different ratings to the K dataset affects the MAE value for different rating values we provide figure 7.3.

Figure 7.3 shows the the MAE value for each rating separately, as well as the overall MAE value for the highest predicted strategy experiment. Overall MAE

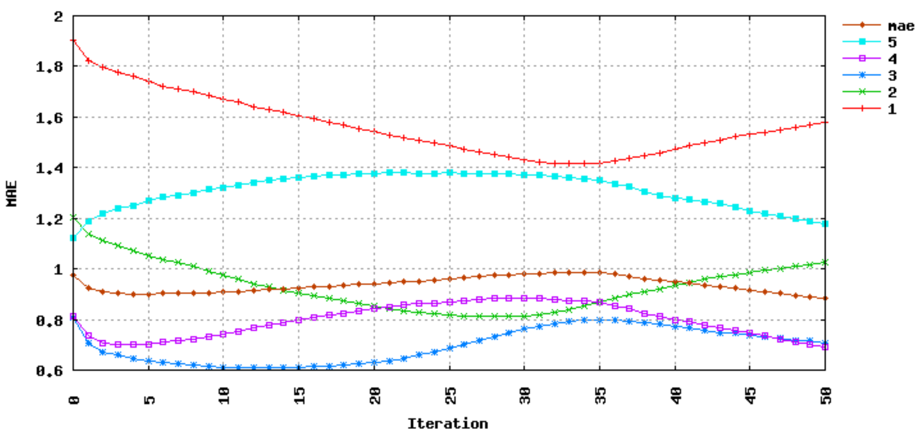

Figure 7.3: MAE layout for the highest predicted strategy experiment

curve is more sensitive to the MAE curves of ratings 3, 4 and 5 than to curves of ratings 1 and 2 because ratings 3, 4 and 5 make up more than 80% of total ratings in the test dataset T.

It is important to notice that adding more high ratings to the training dataset does not result in better prediction accuracy for high ratings. On the contrary, it improves the prediction accuracy for low ratings and worsens the accuracy for high ratings. It results in decreasing overall MAE value in the second stage since high ratings have much more weight in calculating it.

As shown by figures 7.3 and 7.2 the MAE for rating 5 starts to improve only at the at the iteration 30, where the the proportion of items rated 5 becomes more similar to one in the testing dataset.

All the strategies from the second group: highest predicted, popularity, binary predicted and variance add too many high ratings to the K dataset in the first and the second stage and start moving towards restoring the balance of ratings when approaching iteration 30.

It is reasonable to assume that the random and lowest-highest predicted strategies do not introduce so much bias to the training data set, what results in constantly decreasing MAE.

As indicated by figures 7.4 and 7.5 random strategy showed growth rates of ratings which are proportional to the overall distribution of the dataset (figure 6.4). As expected, MAE for all values of ratings were constantly decreasing when random strategy was used, what also resulted in constantly decreasing overall MAE.

Among individual strategies, the random strategy is the best performing strategy with regard to MAE. However, it is important to note that MovieLens dataset has considerably low sparsity. In applications with lower sparsity random strategy would have a lower probability of eliciting items that a user can provide a

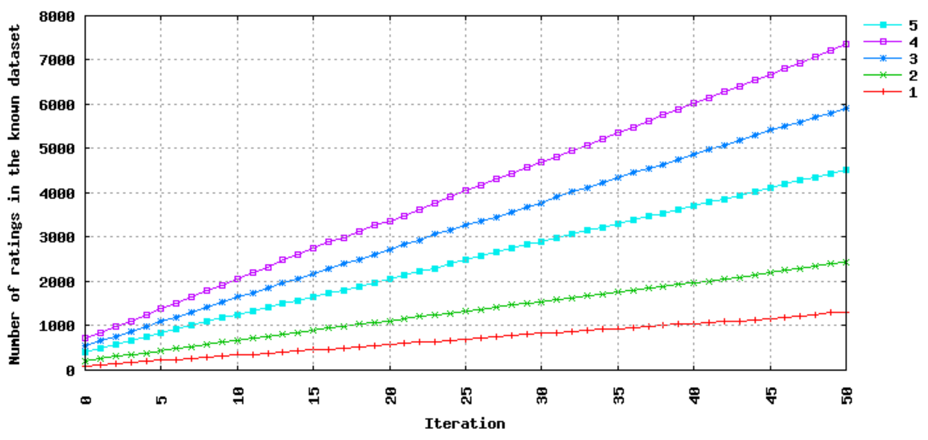

Figure 7.4: Number of different ratings in K dataset depending on the iteration on the process of random strategy experiment

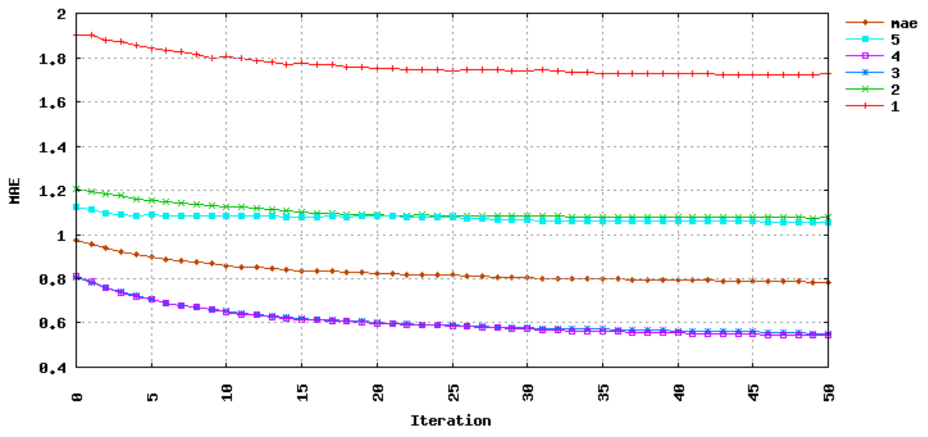

Figure 7.5: MAE layout for the random strategy experiment

rating for.

The good performance of the lowest-predicted strategy with regard to MAE might seem surprising, since it should bias the distribution of ratings in the training database towards low ratings. As shown in figure 7.6, when lowest predicted strategy is used, ratings with values 1, 2 and 3 grow in number a little faster and ratings with value 4 and 5 grow a bit slower than in random strategy.

Results have shown that even if the distribution of ratings becomes slightly unbalanced when adding lowest predicted ratings to the K set, the proportions or rating types are constant and the deviation from the full Movie Lens dataset is not very strong.

One can observe in figure 7.7 that MAE values for ratings 1 and 2 do not improve a lot over the iterations even if the training dataset contains more low value ratings. As it was already observed when discussing highest-predicted strategy,

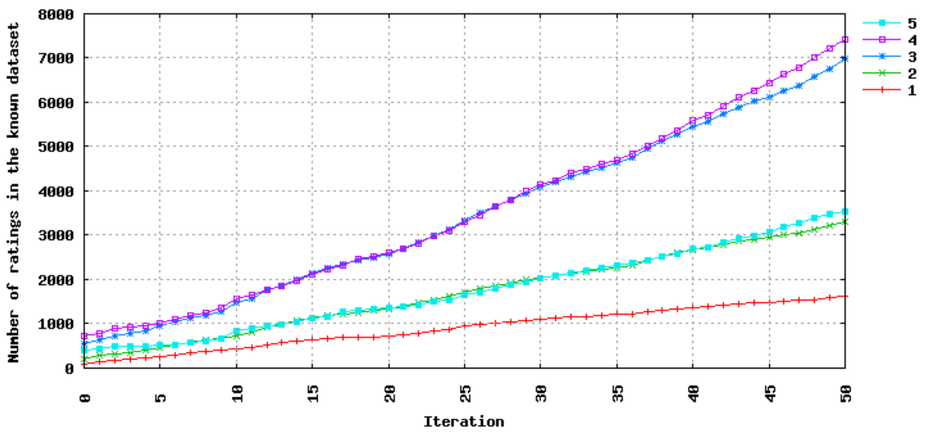

Figure 7.6: Number of different ratings in K dataset depending on the iteration on the process of lowest predicted strategy experiment

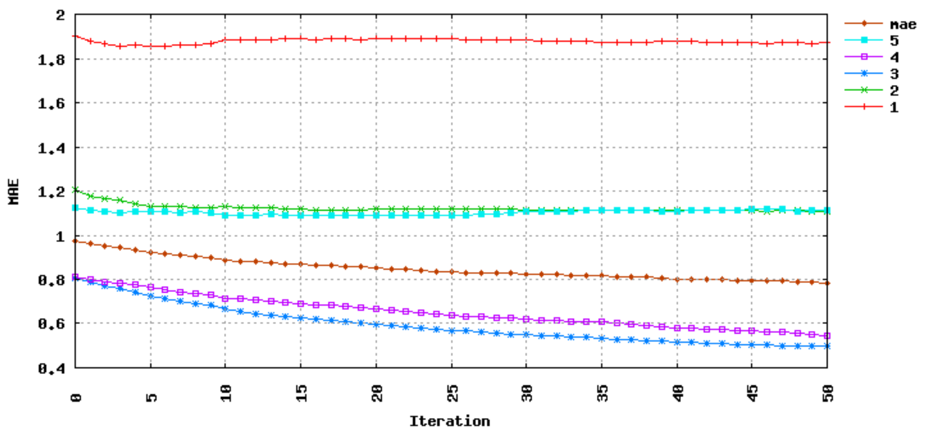

Figure 7.7: MAE layout for the lowest predicted strategy experiment

adding more ratings of a certain type to the training dataset does not improve or even worsens prediction accuracy for that type.

7.1.2 NDCG

In this section we analyse results of the experiments with regard to NDCG metric. As discussed in section 6.3.3 DCG for particular user measures the rating values in the recommendation list generated using predicted rating values. Then normalized discounted cumulative gain (NDCG) is computed by comparing the calculated DCG with with an ideal DCG for that user.

We chose to generate the recommendation lists (ideal and predicted) of length 10 for every user. Recommendations lists are created only from items that have ratings in the testing dataset. In other words, they do not take into account

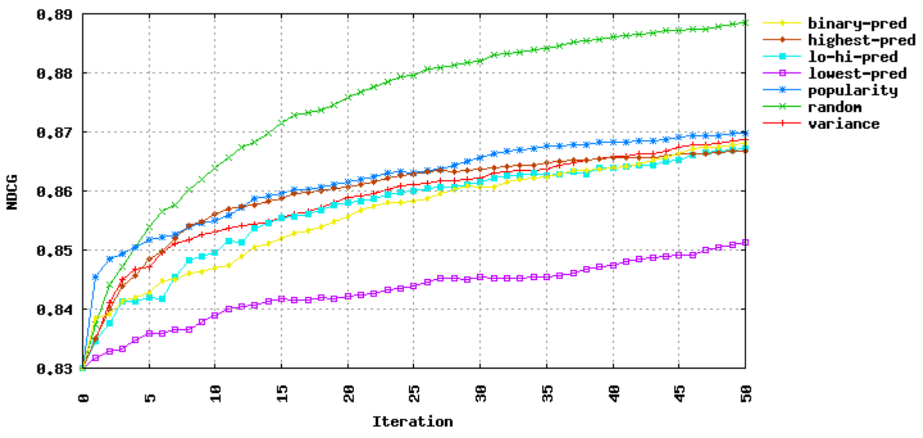

Figure 7.8: NDCG; individual strategies; random split

user-item pairs that don't have a rating in the testing dataset. Sometimes the testing set contains less than 10 items for some users. In this case the ideal recommendation list contains all the items that are in the testing dataset for that user and predicted recommendation list of the same size as the ideal list is used.

Moreover, the testing algorithm is not able to obtain the predicted recommendation list of size 10 sometimes, because, for instance, ratings can be predicted only for 5 items in the testing dataset for a particular user. This happens in cases when the items residing in the testing dataset have no ratings at all in the known dataset K. The prediction algorithm that we use can not derive predictions for items that have no ratings in the training dataset. It is important to notice that ideal recommendation lists for all users stay the same during the experiments that use the same dataset. Therefore, if an algorithm is not able to generate predicted recommendation lists of size 10, lists of the size which is available are used what results in smaller NDCG values.

Figure 7.8 depicts the NDCG curves for the individual strategies. The higher is the NDCG value, the higher rated are the items in the predicted recommendation lists. Popularity is the best strategy in the beginning of the experiment (iterations 1-4). At iteration 4 random strategy overtakes popularity and rises much faster if compared with other strategies. Lowest predicted is by far the worst performing strategy, it stays at the bottom part of the graph for all the duration of the test.

It might seem intuitive to assume that NDCG value depends on the prediction error for high ranked items. This is, however, not true. If we look closely to the MAE layout graphs of random and lowest predicted strategies (figures 7.5 and 7.7 respectively), we can see that prediction error for ratings 5 and 4 is very similar for both strategies and predictions for value 3 are slightly better for lowest predicted. Also, if we compare highest predicted (figure 7.7) and lowest predicted MAE layouts, we can see that predictions for ratings 4 and 5 are better in low predicted strategy for most of the test duration. However, highest predicted performs much better than lowest predicted with regard to NDCG.

It is safe to conclude that NDCG depends a lot more on the prediction algorithm's ability to rank items correctly than on it's ability to predict ratings with little error. We illustrate this statement by an example provided in table 7.1.

Item	i_1	i_2	i_3	i_4	i_5		
Actual Ratings	1	2	3	4	5	NDCG	MAE
Predictions 1	1.1	1.2	1.3	1.4	1.5	1	1.74
Ranks 1	5	4	3	2	1		
Predictions 2	1	2	4.2	4.1	4	0.93	0.46
Ranks 2	5	4	1	2	3		

Table 7.1: NDCG and MAE example

In the example testing dataset there are five items with rating values provided in the 'Actual Ratings' row. Let's assume that we use two different rating prediction algorithms and we use them to obtain predictions that are provided in rows 'Predictions 1' and 'Predictions 2'. Rows 'Ranks 1' and 'Ranks 2' show the ranking of items that were computed using prediction values obtained using the first and the second prediction algorithm respectively. In this example NDCG is computed over three best predicted items. One can see that NDCG value calculated for 'Predictions 1' is 1 (which is the best possible NDCG) and NDCG value calculated for 'Predictions 2' is 0.93, even though MAE value calculated for 'Predictions 1' (1.74) is much higher than the MAE value for 'Predictions 2' (0.46).

To sum up, MAE and NDCG are completely different measures and NDCG only vaguely depends on MAE. In fact, research has been done which addresses item ranking prediction error and proposes a collaborative filtering approach which optimizes ranking prediction [14].

To be able to explain the behaviour of the NDCG curves for different strategies we must first understand the changes of coverage during the test.

For the results provided in 7.8 random split of the dataset was used. Meaning that there were some items and some users which don't have ratings in the known K dataset in the beginning of the test. This models the situation of new users and new items which is common in real recommender systems. Users and items with no ratings might obtain ratings in the process of the test, so it is important to know the coverage values at each iteration of the experiment, which is provided in figure 7.9. Coverage shows what the proportion of ratings in the testing dataset T for whom a rating can be predicted is.

One can see that coverage for binary predicted, highest predicted, lowest-

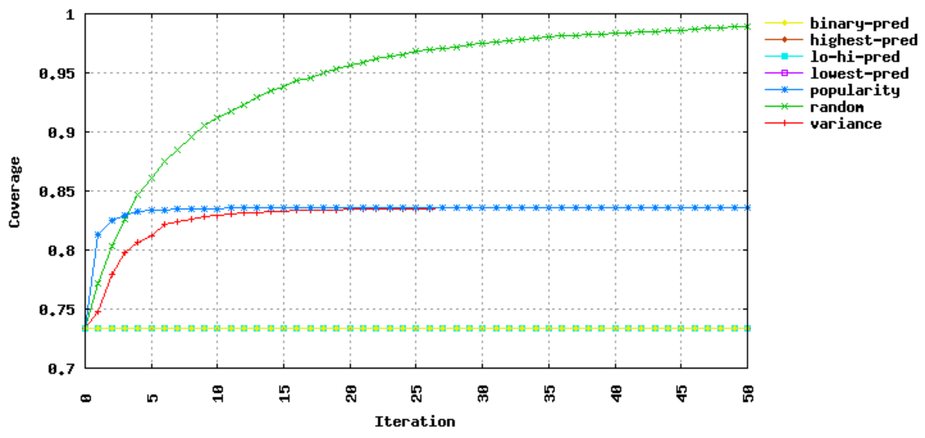

Figure 7.9: Coverage; individual strategies; random split

highest predicted strategies does not change and stays the same during all iterations. This can be explained by the fact that ratings can move from the unknown dataset X to the known dataset K using the aforementioned strategies only in the case when values can be predicted for those ratings. Since the rating prediction algorithm we use is not able to predict ratings for new items and new users, their ratings are never proposed for elicitation and never move to the dataset of known ratings K.

The coverage for popularity and variance strategies rises to approximately 0.84 and stays the same for the rest of the experiment. This is because variance and popularity is calculated with respect to ratings of items (as opposed to ratings for users) and it can only be calculated for those items which have ratings in dataset K. So if there are no ratings for a particular item in K, no ratings for it will ever appear in the K set during the test either. However, even if popularity and variance strategies need at least one rating in K for an item to be able to move ratings for that item from X to K, they do not require the target user to have any ratings. Coverage increases using aforementioned strategies because they are able to elicit ratings for new users. Altough the coverage does not reach 1 because the strategies are not able to elicit ratings for new items.

The coverage for random strategy increases gradually and approaches 1 at the end of the test. Random strategy selects items for elicitation completely randomly. In this way it is able to elicit ratings for both, new users and new items. This way it manages to increase the coverage a lot more than other strategies.

If we come back to analysis of the experimental results with regard to NDCG, we notice that NDCG and coverage graphs bear a particular similarity. In both graphs, NDCG and coverage, popularity performs the best in the beginning of the test and is overtaken by random in the iterations 4 and 5. One can see

Nr. of Ratings in training dataset for User	NDCG	Nr. of Users
< 10	0.91	442
$>= 10$	0.78	497

Table 7.2: NDCG value for different types of users. Testing dataset.

that NDCG and coverage performance correlate for most strategies. Among all strategies, random gains the highest coverage in iteration 4 and the highest NDCG value in iteration 5. Popularity is the second best performing strategy with regard to NDCG on iterations 5-50, although it is shortly overtaken by the highest predicted strategy 9-12 and 26.

Intuitively, the NDCG value should not be affected neither by increasing number of users nor increasing number of items in dataset K. Overall NDCG value is an average of NDCG for each user for whom we can predict at least one rating, that is, we don't take into account users for whom we can not predict ratings and create recommendations.

However, overall NDCG actually depends on the number of users for which we can predict ratings. In fact, experiments showed that NDCG values for newly added users during the test are higher on average than for those users that had ratings in K initially.

In order to understand the reason for this phenomenon it is important to notice that users that have smaller number of ratings in the whole MovieLens dataset are more likely not to appear in the dataset K initially. Assume the following example: user 1 has 20 ratings and user 2 has 100 ratings overall. The split is random, so every rating has an equal probability of p to appear in the dataset K. The probability that no ratings belonging to user 1 will appear in K, therefore, is p^{20}, and the probability that no user's 2 ratings will appear in K is p^{100}. Since $p < 1$, $p^{20} > p^{100}$ i.e. the probability that no ratings of user 1 will appear in K is higher than the probability for user 2.

Actually, users that have low numbers of ratings in K have higher NDCG values than those that have many ratings. A small test has been made on the MovieLens data to check this. Results are provided in table 7.2. Consequently, adding new users results in increasing overall NDCG value.

Our claim that increasing coverage makes a positive effect on NDCG is further supported by the results of the experiment which uses random-all split of MovieLens data described in chapter 6.2 (Figure 7.10). In the aforementioned experiment the data was split in such a way that all items and all users have at least one rating in the initial K dataset. Thus, coverage for all strategies in this case was equal to 1, meaning that ratings for all users and all items could be predicted at any point during the experiment. In this experiment the best performing strategies are highest predicted, popularity and variance. While random and variance performed slightly worse in iterations 0-30 and reached ap-

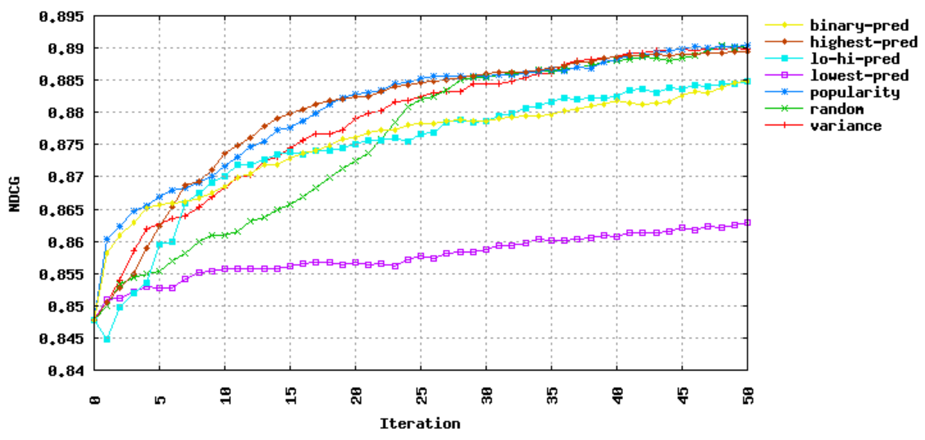

Figure 7.10: NDCG; individual strategies; random-all split

proximately the same NDCG as highest predicted and popularity in iteration 30. Slightly worse performing strategies are variance, binary predicted and low-high predicted. And low predicted strategy shows by far the worst performance as it was in the test which used the random split.

7.1.3 Precision

Precision, as it was described in chapter 6.3.2, measures the proportion of items rated 4 and 5 in the recommendation list. Figure 7.11 depicts the results of experiments that used random split. One can see that highest predicted is the best performing strategy for the most duration of the test. It is overtaken only by binary predicted strategy in the very beginning (iterations 1-3) and in the end (iterations 45-50). Binary predicted and lowest-highest predicted are the second and the third best performing strategies.

Precision values for random strategy, in contrast to the NDCG values, are quite low if compared with highest predicted strategy. This is again related to the fact that coverage increases for random strategy and does not increase for highest-predicted strategy. Increasing coverage for random strategy results in lower overall precision value because it is much lower for new users.

As mentioned in the previous chapter, newly added users are the users that have low numbers of ratings in the MovieLens database and also in the testing dataset. To calculate precision we measure the number of high ratings (4 and 5) in the recommendation lists. Users that have less than 10 ratings in the testing dataset have smaller recommendation lists as well. Therefore, statistically, there are less high ratings in the top recommendation lists of users that have low numbers of ratings in the testing dataset. And on the contrary, users that have many ratings in the testing dataset are more likely to have a large proportion of high ratings in the testing dataset. This is shown in table 7.3 where statistical

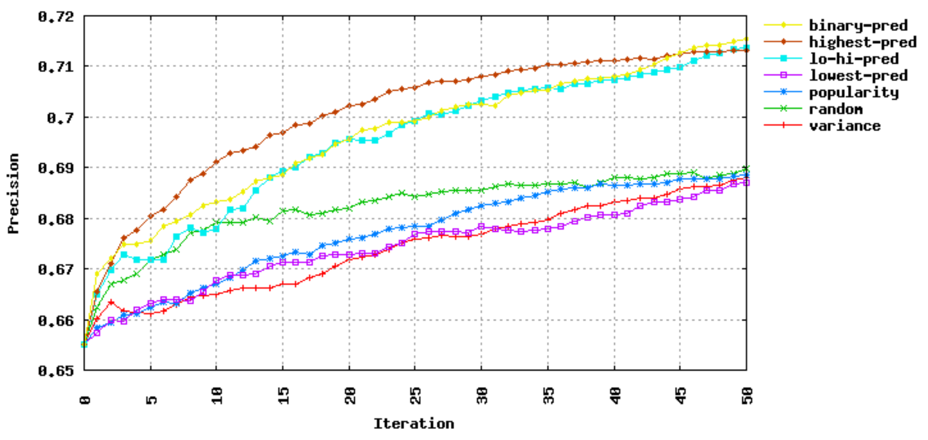

Figure 7.11: Precision; individual strategies; random split

Nr. of Ratings in T for User	5's and 4's in Top 10 (%)	Nr. of Users
< 10	56	223
$>= 10$	91	720

Table 7.3: Numbers of ratings with values 4 and 5 among top 10 ratings for different types of users. Testing dataset.

data for each user was calculated on the testing dataset.

Our assumption is further supported by the results of the experiment that used the random-all split (figure 7.12). When all the items can be found initially in the K dataset and coverage is always equal to 1, random performs slightly better than highest-predicted.

In this experiment, highest predicted precision curve approaches 0.69 at the end of experiment, as opposed to more than 0.71 in the experiment which used random split. This is because the users set in the random-all experiments contains both, users with high and users with low numbers of ratings, as opposed to the experiment which used random split, where users with more ratings were more likely to have ratings in K.

7.2 Partially Randomized Strategies

No individual strategy, except one, is able to elicit ratings for items that have not been evaluated by any user. In this case, when no information is available for the item the only possible strategy is a pure random approach. Partially randomized strategies address this problem by asking new users to rate random items.

On every iteration of the experiment for each user 5 items are selected by the strategy for elicitation. The highest predicted strategy, for instance, when predictions can not be obtained for a new user, is not able to create the elicitation

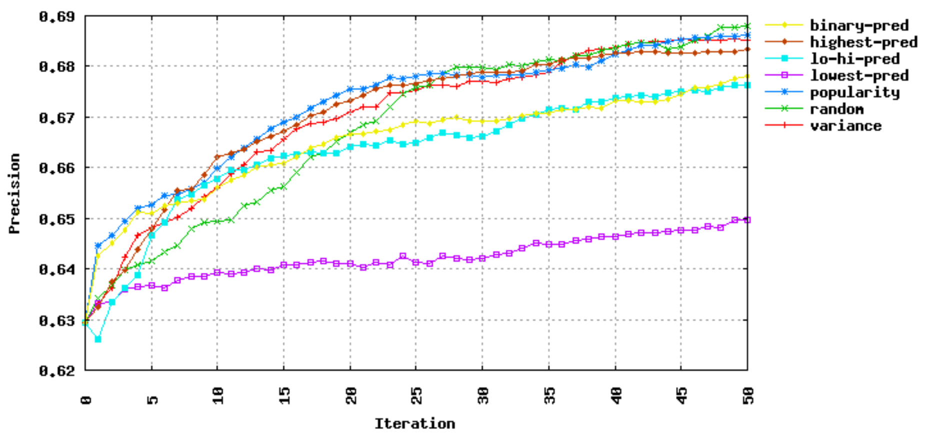

Figure 7.12: Precision; individual strategies; random-all split

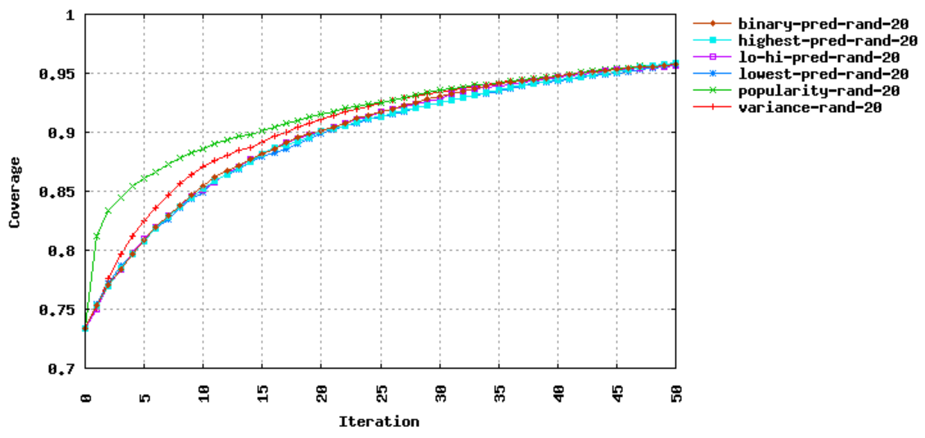

Figure 7.13: Coverage; partially randomized strategies; random split

list. Partially randomized strategies solve this problem by asking a user to rate random items. Moreover, partially randomized strategies modify the elicitation lists to include some random items. This way ratings for new items can be acquired as well.

Figure 7.13 shows coverage change during the experiment. In figures, highest-pred-rand-20, for instance, means that the highest predicted strategy is used with random extension with 20 percent randomization level. It means that if the highest predicted strategy can not create an elicitation list it is created completely randomly, otherwise highest predicted items are elicited substituting 20 percent of the elicitation list with random items. One can see in figure 7.13 that coverage is increasing and gradually approaching 1 for all partially randomized strategies. This is due to the fact that those strategies are able to elicit ratings for both: new items and new users.

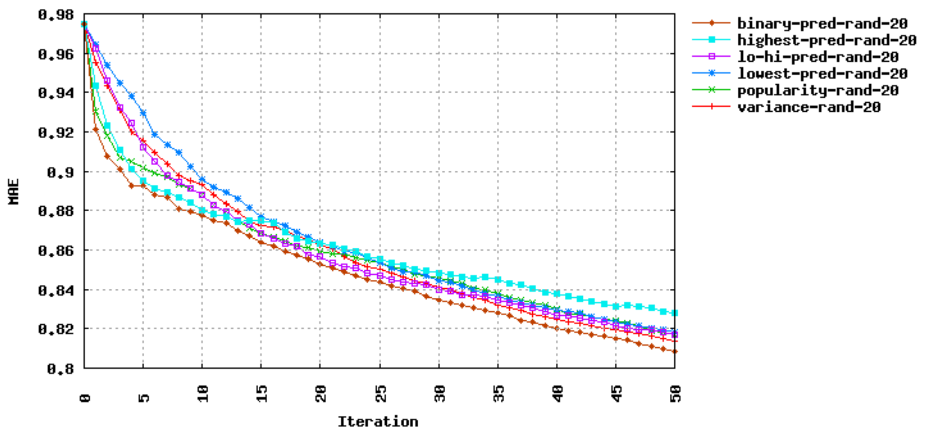

Figure 7.14: MAE; partially randomized strategies; random split

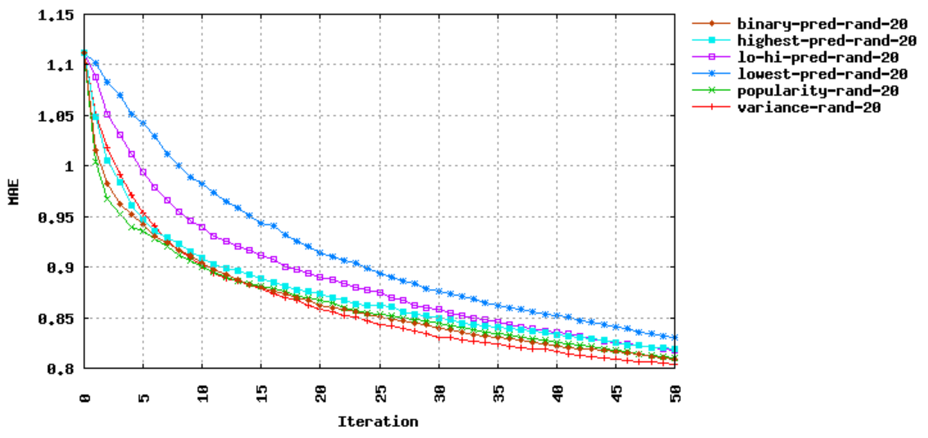

Figure 7.15: MAE; partially randomized strategies; random-all split

Figure 7.14 depicts the MAE change during the experimental process. One can see that all graphs are monotone and there is no such behavior, which was observed on individual strategies (figure 7.1), when in one phase of the experiment the MAE was increasing. Presumably, adding some random items to the elicitation lists reduces the bias of the pure prediction-based strategies. However, comparing the MAE values obtained using the random strategy with the MAE values obtained using the partially randomized strategies, one can see that the former has lower MAE in iterations 6-50. The best of partially randomized strategies with regard to MAE, which is partially randomized binary-predicted, performs better than the pure random only in iterations 1-5.

Different situation was observed in the experimental setup which used the *random-all* split (Figure 7.15). In this setup, where there are no new users and new items, all partially randomized strategies in iteration 50 reach MAE values less than 0.85. Whereas none of individual strategies in the same setup reach 0.85 (figure A.1) using *random-all* split.

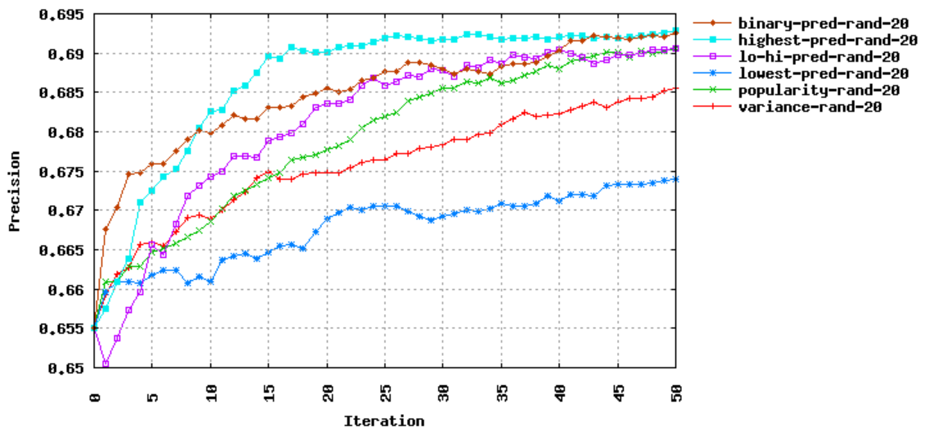

Figure 7.16: Precision; partially randomized strategies; random split

Figures 7.16 and 7.17 show the precision and NDCG values during the experiment. For both metrics the partially randomized highest predicted strategy (hp-rand) shows the best results during all the test except at the beginning.

During the iterations 1-9 the highest precision value is obtained by the the partially randomized binary predicted strategy (bp-rand). And during iterations 1-7 randomly extended popularity strategy (pop-rand) is the best for NDCG.

It is important to note both strategies that show good performance at the beginning are strategies tuned for finding items that a user may know and be able to provide a rating for. Therefore, they are very effective in the beginning when there are many users with very little items in the known dataset K. As shown in figure A.4 bp-rand and pop-rand strategies manage to elicit more items

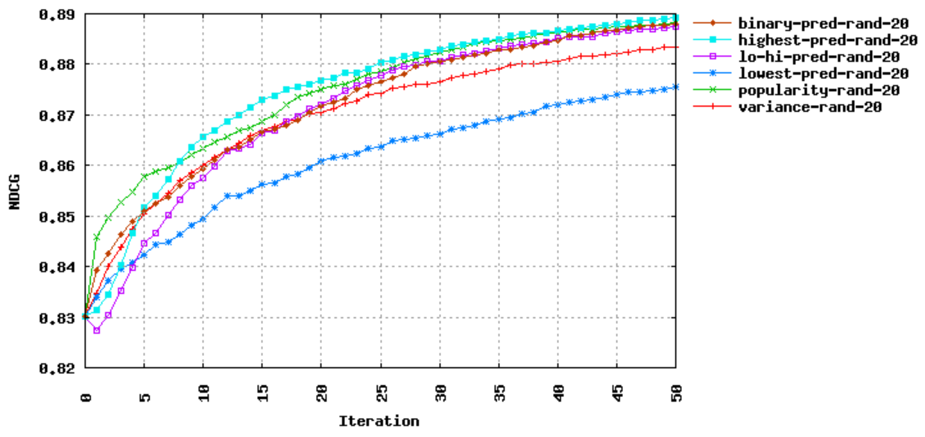

Figure 7.17: NDCG; partially randomized strategies; random split

than other strategies.

Hp-rand strategy also showed very good performance when using random-all split of data (Figures A.2 and A.3). With regard to precision, hp-rand strategy is the best at iterations 11-50. With respect to NDCG, hp-rand is the best at iterations 8-50. As it was observed in the experiment which used random split, strategies emphasizing acquisition of available items work better than other strategies in the starting phase also when the random-all split is used.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this work we addressed the problem of selecting items to ask users about their opinion, which is also defined as the rating elicitation problem. This problem can be solved by giving tools for users to find and rate by themselves an item which they have seen or it can be solved by presenting to them a precomputed list of items which they could rate [20]. The second approach may be more effective in many situations. For instance, when using mobile devices with limited browsing capabilities, where the design should always minimize the human computer interaction.

In our work we dealt with the second problem, where a specific technique (a rating elicitation strategy) is used to select the items that the user should rate. The acquired ratings for those items are supposed to improve the rating prediction accuracy and recommendation quality in the future as much as possible.

In previous work several rating elicitation strategies were proposed and evaluated, such as: asking users to rate items randomly, asking items with the highest rating entropy, most popular items, items with the highest $log(Popularity) * Entropy$ or the personalized item-item approach which asks to rate items similar to the items which a user has already seen [16]. Others [6] proposed item-focused modifications of the techniques presented in [16] to elicit ratings to improve prediction on a specific item. Their results showed that focused strategies are consistently better than unfocused ones with regard to MAE and user effort. Harpale et al. [7] proposed to use a personalized form of Bayesian active learning which combines the entropy minimization and a probability of getting a rating. Results show that their personalized Bayesian method works better with respect to MAE than the standard Bayesian technique.

In our work we have evaluated some strategies that were proposed in previous work [16]: popularity, random, variance. And, in addition, we have also derived some new, prediction-based strategies: binary-prediction, highest-predicted, lowest-predicted, highest-lowest-predicted. Prediction-based strategies select items to be elicited from the user according to their rating predictions. Highest-predicted strategy, for instance, selects items with the highest predicted ratings for a specific user. Binary-predicted strategy tries to predict if the user has consumed a specific item in the past by approximating on a binary (rated/not rated) user-item matrix. Prediction-based strategies are not able to generate items to be presented to users, that don't have any ratings (new users), neither to request rating for items that have no ratings (new items). This happens because predictions can not be made for new users and items with the prediction method that we used.

In addition, we proposed partially randomized strategies, which combine the random strategy with other aforementioned strategies. For instance, highest-predicted is combined with random in such a way that it selects items with the highest predicted ratings and also includes some random items. It also selects random items when the ratings can not be predicted for new users.

To evaluate the strategies we created software which simulates the process of rating elicitation and rating database growth. During the simulation we measured various metrics at different phases of the rating database. The metrics include: MAE to measure the improvements in prediction accuracy, precision to measure the relevance of recommendations, normalized discounted cumulative gain (NDCG) to measure the quality of produced ranking and coverage to measure the proportion of items over which the system can form predictions.

To run the simulations we used the freely available MovieLens rating data. For the experiments we split the data randomly into three groups (ratings known by the system, ratings known by the user but not known by the system and testing data), what created some users and some items that don't have ratings in the initial data known by the system, this way creating a more realistic experimental setup by addressing the problem of new users and new items.

The evaluation showed that different strategies can improve different aspects of the recommendation quality or rating prediction accuracy in different stages of rating database development. Table 8.1 lists the best performing strategies with respect to the chosen measures and the phase of the evaluation. In the starting phase there are many new users and few items. The improvement phase reflects more mature state of the ratings database where, if a strategy managed to address a new user and new item problems, there are very little or no users and items with no ratings.

However, prediction-based strategies neither address the problem of new users, nor of new items. Popularity and variance strategies are able to select items for new users, but can not select items that have no ratings. Partially randomized

	Starting Phase	Improvement Phase
MAE	Random Binary-prediction-rand	Low-high-predicted Lowest-predicted Random
Precision	Highest-predicted	Binary-prediction Lowest-highest-predicted Highest-predicted
NDCG	Highest-predicted-rand Random	Highest-predicted-rand Random Popularity-rand

Table 8.1: The best performing strategies with respect to the simulation stage and various measures

Strategies That Address the Problem of:		
Both, new users and new items	Only new items	None
Random Highest-predicted-rand Lowest-predicted-rand Lowest-highest-predicted-rand Variance-rand Popularity-rand Binary-prediction-rand	Variance Popularity	Highest-predicted Lowest-predicted Lowest-highest-predicted Binary-prediction

Table 8.2: Classification of strategies with regard to their ability to elicit ratings for new users and new items

	Starting Phase	Improvement Phase
MAE	Random Binary-prediction-rand	Random
Precision	Highest-predicted-rand Binary-prediction-rand	Highest-predicted-rand Binary-prediction-rand
NDCG	Highest-predicted-rand Random	Highest-predicted-rand Random Popularity-rand

Table 8.3: The best performing strategies with respect to the simulation stage and various measures (Only strategies that address both, new user and new item problems are included)

strategies solve this problem by selecting random items for users that have no ratings, and also introducing some random items to be presented for users that already have some ratings. The summary of how strategies address the new user and new item problems is provided in table 8.2. We also provide a table 8.3 which summarizes the best performing strategies among those which address both the new item and the new user problems.

We have shown that different strategies can improve different accuracy and quality measures: for instance, the random strategy performs the best with regard to MAE, but it is not the best with regard to precision and NDCG (Table 8.1). We have also shown that the performance of strategy can depend on the stage of rating database development. As for precision, the best strategy in the first phase was highest-predicted, in the second stage it was binary-prediction (Table 8.1). However, some strategies might be the best in all stages of rating database development: for instance, highest-predicted-rand strategy is the best for NDCG in both phases (Table 8.1).

To sum up, the main contributions of our work to this research field are:

- A large set of rating elicitation strategies.
- The evaluation of the strategies behavior with respect to a wide set of metrics (MAE, Precision, NDCG, coverage).
- Some guidelines for the selection of the strategy depending on the rating database size and target evaluation metric.

8.2 Future Work

This research opened a number of new problems that would definetely deserve some more study.

First of all, it is important to note a potential limitation of our research: the results presented in this work clearly depend, as in any study, on the chosen experimental setup, which can partially reflect the real evolution of a recommender system.

In our work we assume that a randomly chosen set of ratings, among those that the user really gave to the system, represents the ratings known by the user, but unknown by the system. However, this set does not completely reflect all the user knowledge, it contains only the ratings acquired using the specific MovieLens recommender, which used a combined random and popularity technique for rating elicitation.

In future work, it would be useful to conduct the same experiments using other datasets. In particular it could be useful to test the proposed strategies exploiting the rating submission time. It would allow us to model users more precisely i.e. it would be possible to split the rating data according to time. However, even this approach has many limitations. For instance, the rating submission time does not really tell us the time when the item was consumed. Meaning, that it would be still impossible to tell whether at a certain point of time a user could have provided a rating for a particular item or not.

It is also important to note that performance of some strategies (random, for instance) depend on the sparsity of rating data. MovieLens data, which was used in our experiments, has a considerably low sparsity compared to other datasets, which may have influenced the results of our simulation. For example, if the data sparsity was higher, there would be only a very low probability for random strategy to select a rating that a user has consumed in the past and can provide a rating for.

Furthermore, there remain many unexplored possibilities for combining strategies. There lies a great potential in rating elicitation strategies that use different approaches depending on the state of the target user. For instance, asking users to rate popular items when a user does not have any ratings yet and using another strategy at a latter stage.

It would be also beneficial to run the same experiments on the best strategies proposed in previous works [16] [6] and compare them with the performance of the strategies proposed in our work. Moreover, the possibility of selecting items according to factors of availability and usefulness was not covered in this thesis. The future research in the area could propose a strategy which combines a probability that a user has used an item and thus can provide a rating for it with the usefulness of gaining that item.

Bibliography

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, June 2005. [cited at p. 3]

[2] C. Anderson. *The Long Tail.* Random House Business, 2006. [cited at p. 3]

[3] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. pages 43–52. Morgan Kaufmann, 1998. [cited at p. 4, 10]

[4] Robin Burke. Knowledge-based recommender systems, 2000. [cited at p. 4]

[5] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002. [cited at p. 4, 9]

[6] Giuseppe Carenini, Jocelyin Smith, and David Poole. Towards more conversational and collaborative recommender systems, 2003. [cited at p. 4, 5, 14, 17, 18, 47, 51]

[7] Abhay S. Harpale and Yiming Yang. Personalized active learning for collaborative filtering. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 91–98, New York, NY, USA, 2008. ACM. [cited at p. 4, 5, 18, 47]

[8] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004. [cited at p. 26, 28]

[9] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. [cited at p. 10]

[10] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002. [cited at p. 26]

[11] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, 1997. [cited at p. 10]

[12] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, NY, USA, 2008. ACM. [cited at p. 4, 11]

[13] Ken Lang. Newsweeder: Learning to filter netnews. In *in Proceedings of the 12th International Machine Learning Conference (ML95*, 1995. [cited at p. 3]

[14] Nathan N. Liu and Qiang Yang. Eigenrank: a ranking-oriented approach to collaborative filtering. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 83–90, New York, NY, USA, 2008. ACM. [cited at p. 27, 38]

[15] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266, New York, NY, USA, 2003. ACM. [cited at p. 23]

[16] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. Mcnee, Joseph A. Konstan, and John Riedl. Getting to know you: Learning new user preferences in recommender systems. pages 127–134. ACM Press, 2002. [cited at p. 4, 5, 6, 16, 17, 18, 47, 48, 51]

[17] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *CSCW*, pages 175–186, 1994. [cited at p. 4, 10]

[18] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997. [cited at p. 3]

[19] Elaine Rich. User modeling via stereotypes. *Cognitive Science*, 3(4):329–354, 1979. [cited at p. 4]

[20] Joseph A. Konstan Sean M. McNee, Shyong K. Lam and John Riedl. Interfaces for eliciting new user preferences in recommender systems. In *User Modeling 2003*, 2003. [cited at p. 4, 5, 18, 47]

[21] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. [cited at p. 10]

[22] LLC Timely Development. Netflix prize, 2008. http://www.timelydevelopment.com/demos/NetflixPrize.aspx. [cited at p. 11]

[23] B. Webb. Netflix update: Try this at home, 2006. http://sifter.org/ simon/journal/20061211.html. [cited at p. 11]

[24] Markus Weimer, Alexandros Karatzoglou, and Alex Smola. Adaptive collaborative filtering. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 275–282, New York, NY, USA, 2008. ACM. [cited at p. 27]

[25] Ming Yi and Weihua Deng. A utility-based recommendation approach for e-commerce websites based on bayesian networks. *Business Intelligence and Financial Engineering, International Conference on*, 0:571–574, 2009. [cited at p. 4]

Appendices

Appendix A

Additional Figures

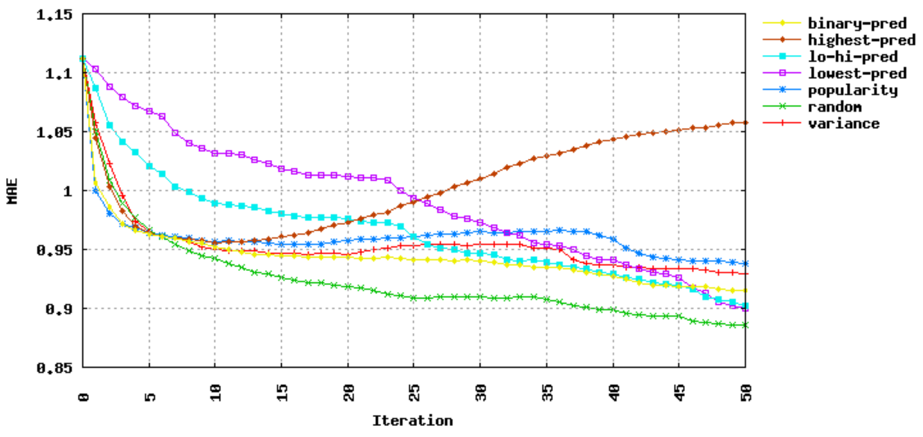

Figure A.1: MAE; individual strategies; random-all split

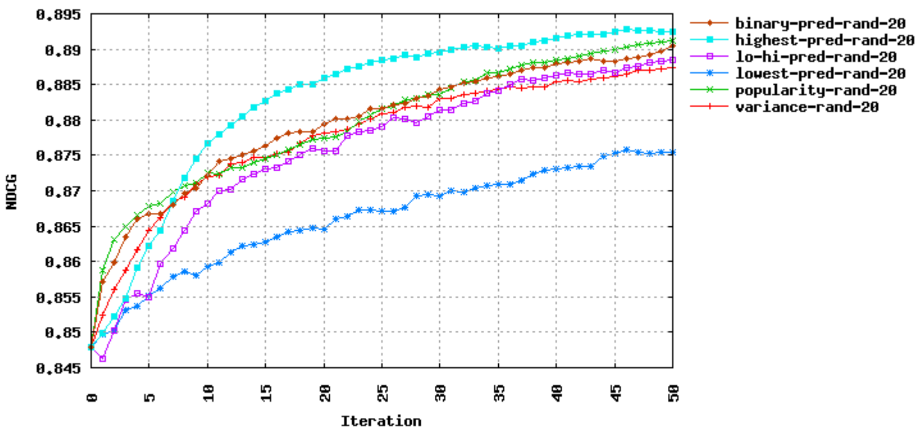

Figure A.2: NDCG; partially randomized strategies; random-all split

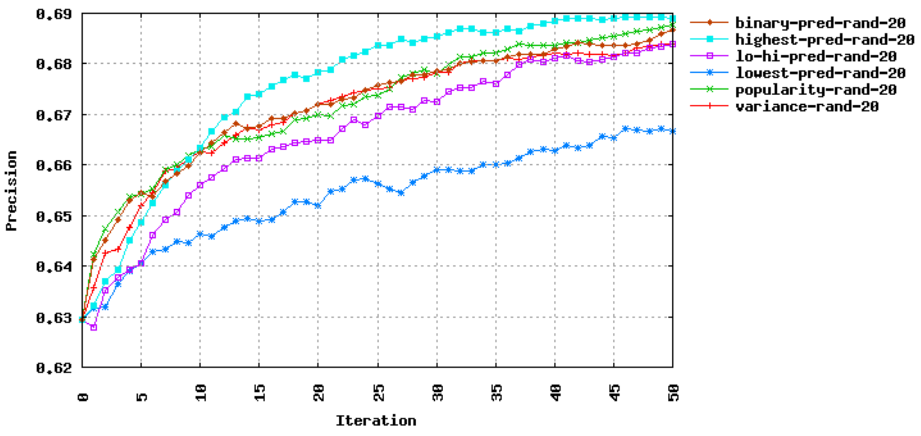

Figure A.3: Precision; partially randomized strategies; random-all split

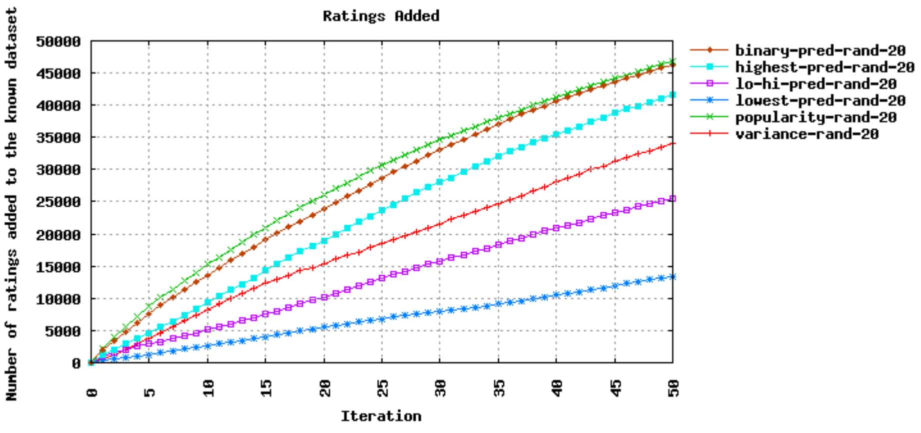

Figure A.4: Ratings Added; partially randomized strategies; random split

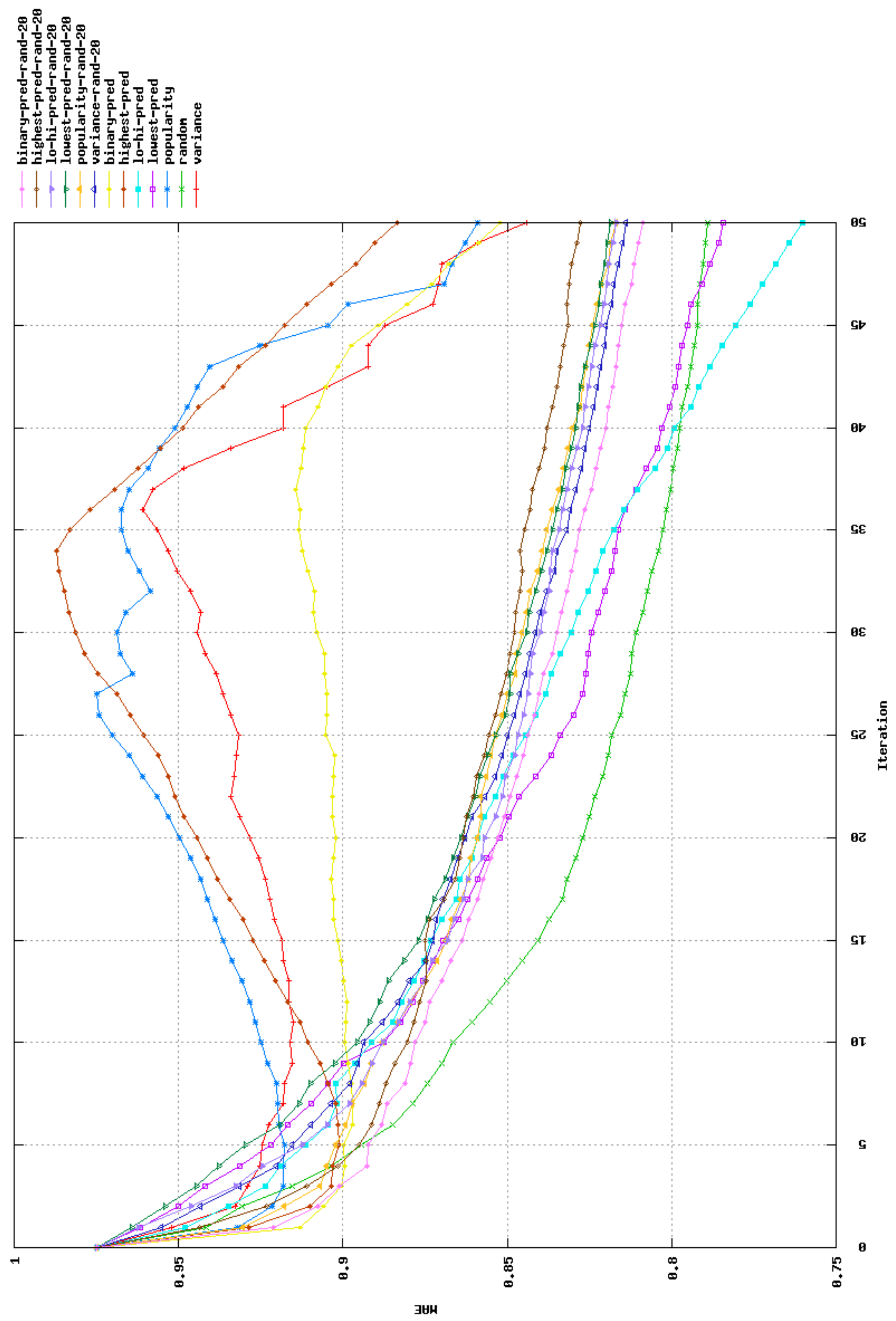

Figure A.5: MAE; all strategies; random split

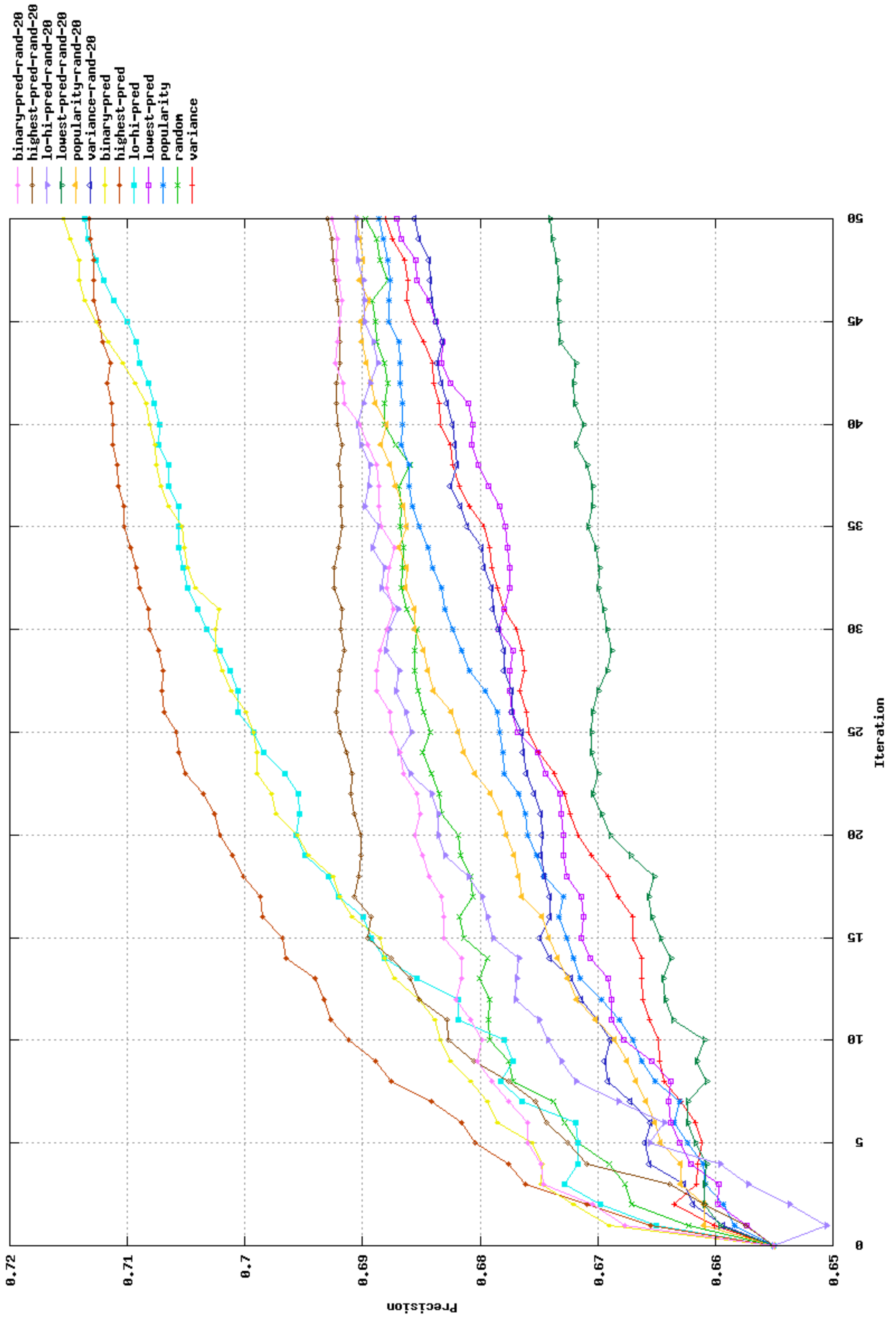

Figure A.6: Precision; all strategies; random split

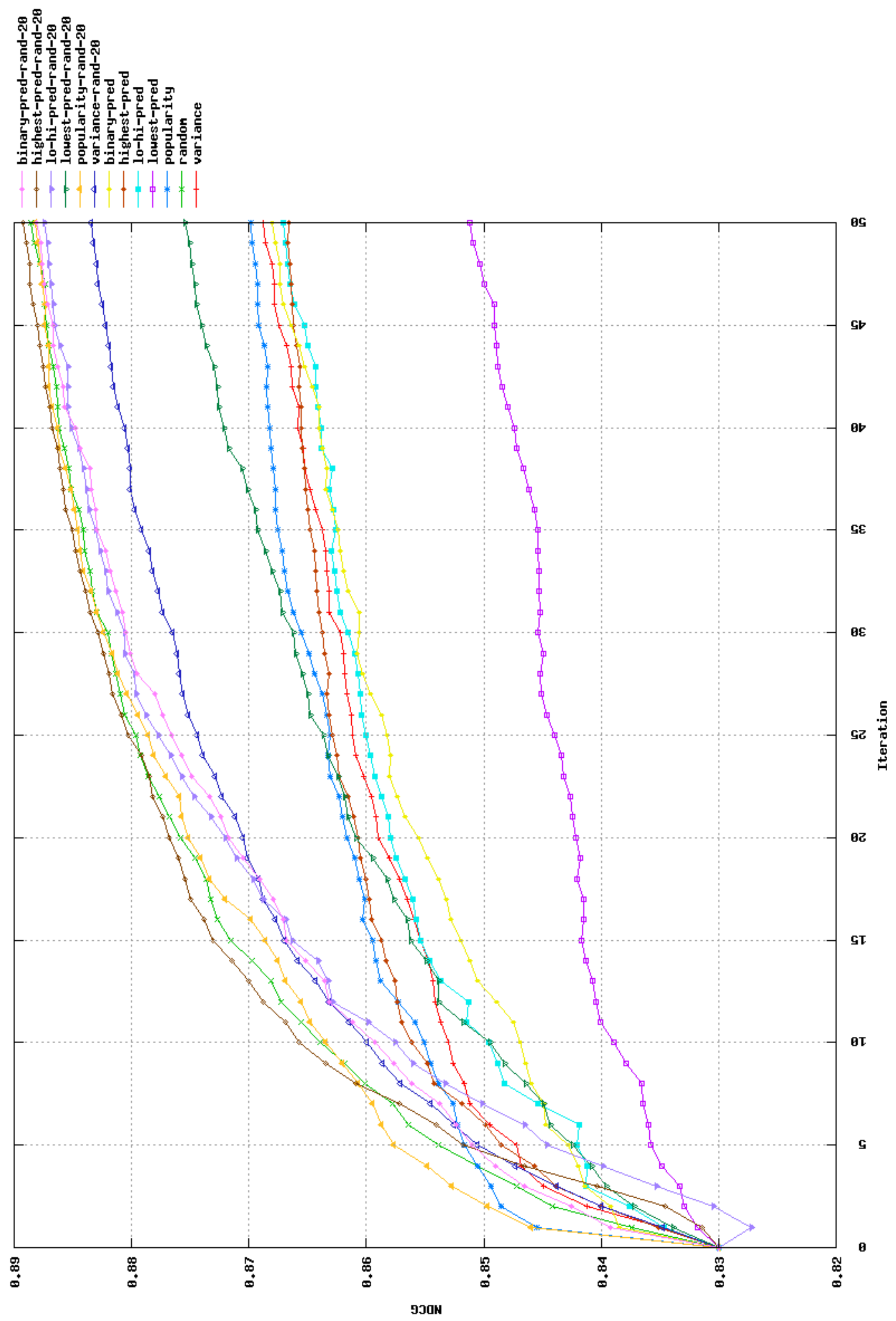

Figure A.7: NDCG; all strategies; random split

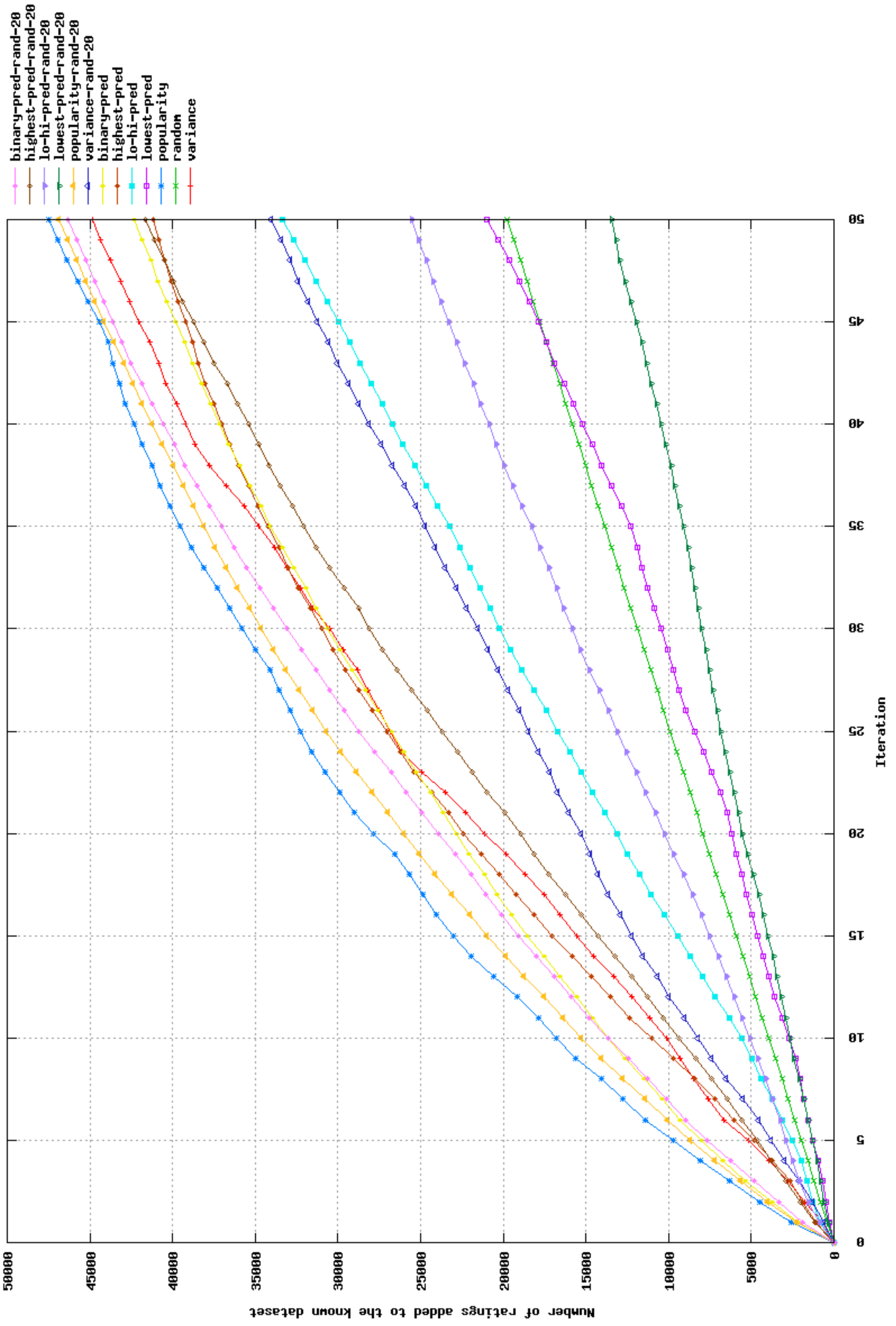

Figure A.8: Ratings Added; all strategies; random split

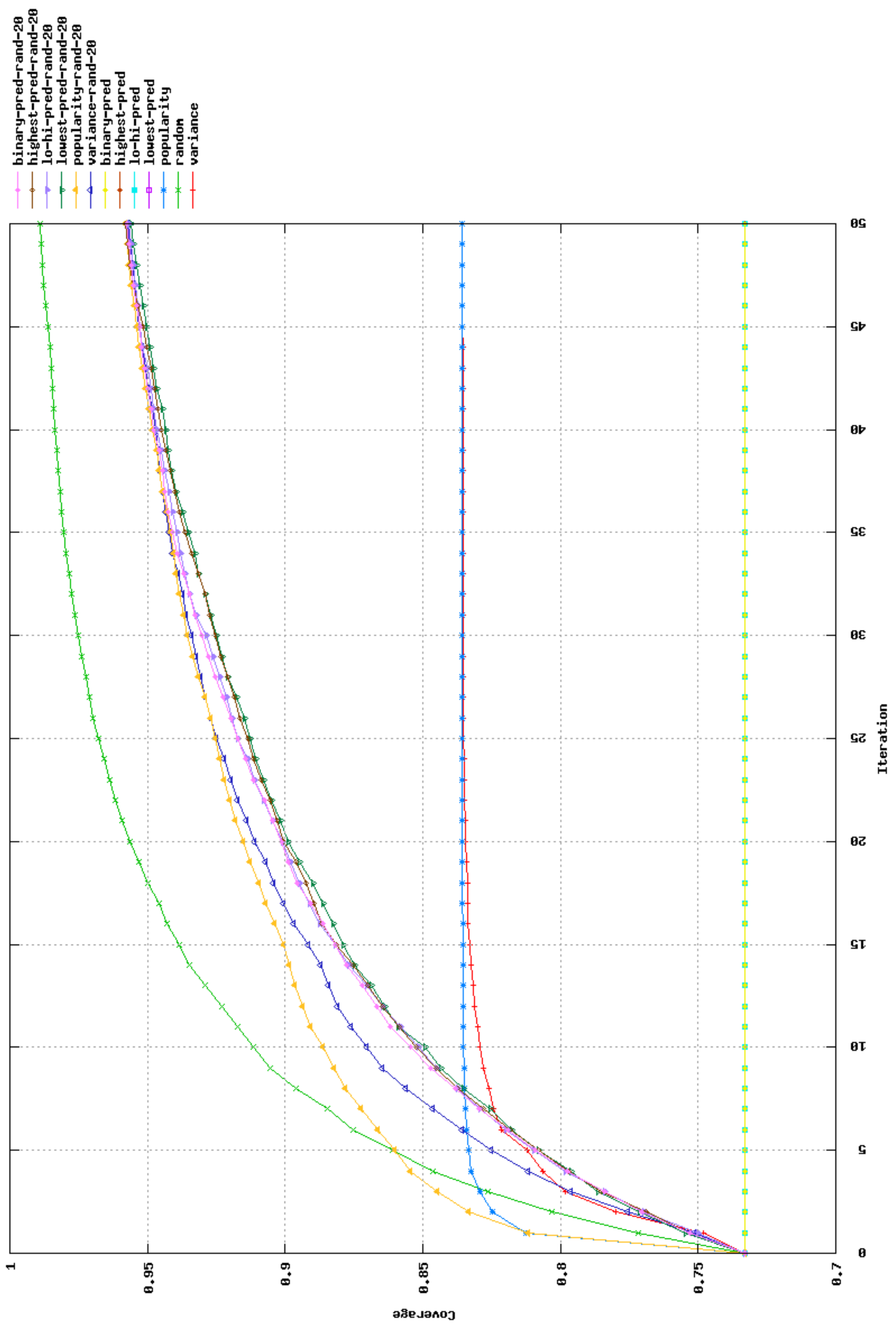

Figure A.9: Coverage; all strategies; random split