RF-REC: Fast and Accurate Computation of Recommendations based on Rating Frequencies

Fatih Gedikli, Faruk Bağdat, Mouzhi Ge, and Dietmar Jannach
Department of Computer Science, 44221 Dortmund, Germany
Email: firstname.lastname@tu-dortmund.de

***Abstract*—The goal of recommender systems (RS) is to provide personalized recommendations of products or services to users facing the problem of information overload on the Web. The most popular approaches to retrieve the most relevant items for a user are collaborative filtering (CF) recommendation algorithms and in particular in recent years a number of sophisticated algorithms based, e.g., on matrix factorization or machine learning, have been proposed to improve the predictive accuracy of RS.**

In our recent work, we proposed a novel recommendation scheme called RF-REC, which generates predictions simply by counting and combining the frequencies of the different rating values in the usual user-item rating matrix. The scheme has some key advantages when compared with more sophisticated techniques. It is trivial to implement, can generate predictions in constant time and has a high prediction coverage.

In this paper we propose extensions to our method in order to further increase the predictive accuracy by introducing schemes to weight and parameterize the components of the predictor. An evaluation on three standard test data sets reveals that the accuracy of our new schemes is higher than traditional CF algorithms in particular on sparse data sets and on a par with a recent matrix factorization algorithm. At the same time, the key advantages of the basic scheme such as computational efficiency, scalability, simplicity and the support for incremental updates are still maintained.

***Keywords*-recommender systems, collaborative filtering, accuracy, computational efficiency, evaluation**

I. INTRODUCTION

Due to the considerable growth of online information, it became a constant challenge to help Internet users to deal with the corresponding information overload. Over the last decade, various techniques in the areas of information retrieval (IR) and information filtering have been developed to help users find items that match their information needs and filter out unrelated information items. In contrast to information filtering techniques implemented in search engines, whose aim is to retrieve the desired information from a large amount of information based on a user query, recommender systems (RS) are today commonly in use on e-commerce platforms to help online visitors finding relevant information or items to purchase in a personalized way.

When applied in the context of e-commerce, the aim of recommender systems is to provide personalized recommendations that best suit a customer's taste, preferences or individual needs. The advantages of using recommender systems are manifold. They can for example help to build better relationship with customers, increase the value of e-business or broaden sales diversity ([1], [2], [3]). In practice, recommender systems have been implemented in different domains such as tourism, entertainment, or book sales, see, for example, [4], [5], or [6].

Although recommender systems have their roots in IR [7], from the mid-1990s recommender systems have become an independent research area of its own. Commonly, recommender systems are classified into four categories: collaborative filtering, content-based filtering, knowledge-based systems and hybrid recommendation approaches[1]. Collaborative filtering (CF) approaches exploit the wisdom of the crowd and recommend items based on the similarity of tastes or preferences of a larger user community. Content-based approaches, on the other hand, recommend items by analyzing their features to identify those items that are similar to the ones that the user preferred in the past. Knowledge-based recommender systems, finally, rely on explicit user requirements and some form of means-ends knowledge to match the user's needs with item characteristics. In order to benefit from the advantages of the different main approaches, hybrid recommendation systems try to combine different algorithms and exploit information from various knowledge sources. Studies have shown that for example hybrids which combine content-based and collaborative filtering can lead to more accurate predictions than pure CF or content-based recommenders [10], [11].

From the four categories above, collaborative filtering is considered to be one of the most successful technologies in practice [8]. Over the last decade, a variety of collaborative filtering algorithms have been proposed in both academia and industry. Boosted by the 1 million dollar Netflix Prize competition[2], in particular in the last few years a variety of sophisticated CF algorithms have been proposed, which rely, e.g., on matrix factorization, probability theory and advanced machine learning techniques [12], [13].

In our previous work, we proposed a novel recommendation scheme called RF-REC [14], which in contrast to other CF approaches bases its rating prediction for a user u and

[1]See [8] or [9] for recent overviews.
[2]http://netflixprize.com

an item i not yet seen by u solely on the information on frequencies of rating values in the database. If, for example, the most frequent rating for item i in the database is 4 and user u's most frequent rating is also 4, the algorithm will basically also predict 4 as u's rating for i.

The main advantages of the scheme in contrast to other algorithms are that (a) no computationally expensive model-building phase is required and predictions can be made in constant time, (b) that the implementation is trivial, and (c) that the ratings that newly arrive in the database can be easily incorporated. First experiments on commonly used evaluation data sets showed that RF-REC's predictive accuracy is comparable or better than the one of traditional nearest-neighbor CF algorithms when using the Mean Absolute Error as a metric [14].

In this work we propose extensions for the basic scheme to increase the recommendation accuracy of RF-REC up to the level of recently proposed algorithms based on matrix factorization. At the same time, our extensions are designed in a way that the key advantages of simplicity and learning- and run-time efficiency are maintained.

The paper is organized as follows. In the next section, we characterize the general CF prediction problem, introduce the RF-REC scheme from [14] and present two new extensions to it. Afterwards, we describe the setting of our experimental evaluation, that is, the used data sets, the evaluation procedure and metrics as well as the algorithms with which we compare our new techniques. After the discussion of the findings in the subsequent section, the paper ends with a summary and an outlook on future work.

II. COLLABORATIVE FILTERING WITH RF-REC

The RF-REC prediction scheme for collaborative filtering from [14] can be summarized as follows.

A. *The Basic* RF-REC *Scheme*

In collaborative filtering approaches, the only input to the recommender is a matrix, which holds the ratings given to the items by a set of users. Table I shows such a matrix, in which the ratings are given on a one-to-five scale, where 5 denotes the best possible rating. The task of a CF recommender consists of producing a list of items that the current user of the system has not seen yet and will most probably like. To that purpose, recommenders predict ratings for unseen items and often create the recommendation list by picking the n top-rated items from the list.

Let us illustrate the basic RF-REC scheme based on the problem of predicting the Alice's rating for item $I3$. The basic rationale of RF-REC is to consider the rating *frequencies* of both the target user[3] Alice and the target item $I3$. As can be seen in Table I, observing the past rating behavior of Alice reveals that she is likely to rate items

[3]We denote the user and item for which a recommendation is sought for as target user and target item respectively.

Table I
EXAMPLE USER-ITEM RATING MATRIX.

	$I1$	$I2$	$I3$	$I4$	$I5$	$\overline{user}$
$Alice$	1	1	**?**	5	4	2.75
$U1$	2		5	5	5	4.25
$U2$			1	1		1.00
$U3$		5	2		2	3.00
$U4$	3		1	1		1.67
$U5$	1	2	2		4	2.25
$\overline{item}$	1.75	2.67	2.20	3.00	3.75	

with the lowest rating value of 1. She rated 2 out of 4 items with 1 and her average rating value is rather low (2.75). On the other hand, if we look at the target item's rating frequencies, we will see that low rating values were assigned to item $I3$. Actually, the rating values 1 and 2 dominate in the rating vector of item $I3$. The frequency-based RF-REC recommendation scheme would therefore predict "1" as Alice's rating for item $I3$ because this rating value has a dominant position in the rating vectors of both the target user and the target item.

Formally, the prediction function for a given user u and an item i in the RF-REC recommendation scheme is defined as follows [14]:

$$\hat{r}_{u,i} = \arg\max_{v \in \mathcal{R}} \left(freq_{user}(u,v) + 1 + \mathbf{1}_{user}(u,v)\right) \cdot \left(freq_{item}(i,v) + 1 + \mathbf{1}_{item}(i,v)\right) \quad (1)$$

where $freq_{user}$ is the frequency of ratings with value v of the target user u and $freq_{item}$ is the frequency of ratings with value v of the target item i. $\mathcal{R}$ corresponds to the set of all possible rating values, e.g., $\mathcal{R} = \{1,2,3,4,5\}$ when using a 5-point rating scale. $\mathbf{1}_{user}(u,v)$ and $\mathbf{1}_{item}(i,v)$ are indicator functions, which return 1 if the given rating corresponds to the rounded average rating of the target user or target item respectively and 0 otherwise. The "1" in the middle of the formula is used to avoid that in situations in which a user has never given a rating (or an item never received a particular rating), the whole term is multiplied with zero. The indicator function shall help in situations, in which several ratings have the same frequency counts. If this is the case and in addition one of these ratings corresponds to the average rating, some small extra weight is added to it so that the average rating is preferred slightly.

In the example in which the frequency of ratings of Alice are [1:2, 2:0, 3:0, 4:1, 5:1] and the rating frequencies of item $I3$ are [1:2, 2:2, 3:0, 4:0, 5:1], we would calculate:

Rating value 1: $(2+1+0)\cdot(2+1+0) = 9$
Rating value 2: $(0+1+0)\cdot(2+1+1) = 4$
...
Rating value 5: $(1+1+0)\cdot(1+1+0) = 4$

Since the formula result for rating 1 is the highest, we predict that Alice will assign the rating value "1" to item $I3$, i.e., Alice will dislike item $I3$. Note that in contrast to traditional CF approaches, RF-REC does not compute rating averages of neighbors and in some sense therefore also has a slight higher tendency to produce also "extreme" ratings.

If we denote the user and item component of our RF-REC scheme by $f_{user}(u, v)$ and $f_{item}(i, v)$ respectively, we can rewrite Equation (1) as

$$\hat{r}_{u,i} = \arg\max_{v \in \mathcal{R}} \; f_{user}(u, v) \cdot f_{item}(i, v) \tag{2}$$

In the rest of this paper we will refer to Equation (2) as the *basic* RF-REC *scheme*. We will now describe two extensions to our frequency-based recommendation scheme.

B. The WEIGHTED RF-REC Scheme

One of the drawbacks of the basic RF-REC scheme is that the prediction value $\hat{r}_{u,i}$ is always an integer value $v \in \mathcal{R}$, i.e., RF-REC returns only such integer values and thus loses precision by ignoring the existing, more fine-grained information contained in the data. In some cases this behavior can lead to poor results. Consider, for example, the case where RF-REC cannot really decide between two rating values, e.g., 2 and 3, because of similar frequency levels. In this case, RF-REC would select a rating value randomly and return either 2 or 3 as the prediction. Intuitively, however, a rating value of 2.5 would be a far better prediction.

With the WEIGHTED RF-REC scheme which we propose next, we suggest to use the rating frequencies also as additional weights for the rating values. We define the WEIGHTED RF-REC scheme as the following weighted average:

$$\hat{r}_{u,i} = \frac{\sum_{v \in \mathcal{R}} f_{user}(u, v) \cdot f_{item}(i, v) \cdot v}{\sum_{v \in \mathcal{R}} f_{user}(u, v) \cdot f_{item}(i, v)} \tag{3}$$

Compared to the basic RF-REC scheme which only puts emphasis on *one* rating that maximizes the product in Equation (2), the weighted scheme considers the rating frequencies of all ratings for computing a prediction.

C. The PARAMETERIZED RF-REC Scheme

In this scheme we apply the idea of weighting the rating values by their frequencies on the user- and item-model independently. The weighted user-model $\hat{r}^{user}_{u,i}$, for example, can be defined as follows:

$$\hat{r}^{user}_{u,i} = \frac{\sum_{v \in \mathcal{R}} f_{user}(u, v) \cdot v}{\sum_{v \in \mathcal{R}} f_{user}(u, v)} \tag{4}$$

The idea applies analogously to the item-model $\hat{r}^{item}_{u,i}$.

With the PARAMETERIZED RF-REC scheme we propose a model which computes a weighted sum of the user- and item-models leading to:

$$\hat{r}_{u,i} = w_u \cdot \hat{r}^{user}_{u,i} + w_i \cdot \hat{r}^{item}_{u,i} \tag{5}$$

The parameters w_u and w_i are weights controlling the influence of the corresponding models.

In order to estimate the user-dependent parameters w_u and the item-dependent parameters w_i, we can solve the corresponding least squares problem:

$$\min_{w_*} \sum_{(u,i) \in \mathcal{K}} (r_{u,i} - \hat{r}_{u,i})^2 + \lambda(\sum_u w_u^2 + \sum_i w_i^2)$$

With the first term $\sum_{(u,i) \in \mathcal{K}} (r_{u,i} - \hat{r}_{u,i})^2$ we try to find optimal parameter settings such that the prediction error $r_{u,i} - \hat{r}_{u,i}$, which plays an important role for accuracy metrics (see Section III-C), is minimized. The last term $\lambda(\sum_u w_u^2 + \sum_i w_i^2)$ is a regularization term with a regularization factor λ which is introduced to avoid overfitting of the parameters to the training examples $\mathcal{K}$ by penalizing them accordingly.

Such minimization problem can be solved by using least square solvers available in different linear algebra packages. However, in various recent recommendation approaches as for example in [12], *gradient descent* is used as a method for optimizing algorithm parameters for a given data set because of its computational efficiency. For this reason we also applied a gradient descent solver in our work to find optimal parameter values w_u and w_i for Equation (5). Algorithm 1 shows the pseudo code of the gradient descent scheme, which is executed once in an offline phase. Note that these parameters have to be learned only once (or only occasionally when the data set significantly changes over time) and that the optimization procedure has not to be re-executed when new ratings are added to a running recommender.

Algorithm 1 Learn parameters with gradient descent solver.

Require: $\#iterations$, γ, λ
  // Gradient descent sweeps:
  for 1 to $\#iterations$ **do**
    for each user u **do**
      for each rated item i of user u **do**
        $\hat{r}_{u,i} \leftarrow w_u \cdot \hat{r}^{user}_{u,i} + w_i \cdot \hat{r}^{item}_{u,i}$
        $e_{u,i} \leftarrow r_{u,i} - \hat{r}_{u,i}$
        // Perform gradient step on w_u*:*
        $w_u \leftarrow w_u + \gamma \cdot (e_{u,i} - \lambda \cdot w_u)$
        // Perform gradient step on w_i*:*
        $w_i \leftarrow w_i + \gamma \cdot (e_{u,i} - \lambda \cdot w_i)$
      end for
    end for
  end for
  return w_u for each user u *and* w_i for each item i

The algorithm iterates over all available ratings in the training data $\mathcal{K}$. The prediction error $e_{u,i}$ represents the gradient in our case. The idea is to change the parameter values w_* by moving into the opposite direction of the gradient.

The movement is controlled by the meta-parameters γ and λ, where γ describes the step-size and λ is the regularization factor described above. The optimal settings for the meta-parameters were determined in additional experiments.

III. Experimental setup

In order to evaluate the predictive accuracy of our new schemes, we compared them to different existing algorithms using standard data sets as well as common evaluation procedures and metrics.

A. Data sets

We evaluated the RF-Rec schemes on three data sets from the movie domain. We used two popular MovieLens data sets[4] – the 100k and the 1M-MovieLens rating database – and a snapshot of the Yahoo!Movies data set[5]. Table II shows the characteristics of the data sets. Note that both MovieLens data sets only contain users who have rated at least 20 items, whereas the minimum number of rated items per user in the Yahoo!Movies data set is 10.

Table II
Data set characteristics.

Data set	$\|U\|$	$\|I\|$	Ratings	Sparsity
100k-MovieLens	943	1,682	100,000	0.9369
1M-MovieLens	6,040	3,900	1,000,209	0.9575
Yahoo!Movies	7,642	11,915	211,231	0.9976

B. Baseline algorithms used for comparison

We compare our new schemes with the following baseline algorithms commonly used in literature.

- *Per User Average*. One straightforward way to calculate a rating prediction is to take the average of the previous ratings of this target user. Since this basic algorithm was often used as a baseline scheme in collaborative filtering works in the past, we also include this technique in our evaluation.
- *Non Personalized/Bias from Mean*. The idea of this also comparably trivial method is to calculate the average deviation from the user's mean for the target item across all users and add this "bias" to the user average ([15], [16]).
- *Weighted Slope One*. Slope One is a family of prediction schemes proposed by Lemire and Maclachlan in [15]. Similar to our work, it was designed to be an algorithm which is comparably easy to implement, supports the incorporation of new ratings, has a reasonable accuracy and is efficient at run-time. Slope One predictors have the form $f(x) = x + b$ and are based on rating differences between items. Weighted Slope One is a variant that weighs the numbers of observed ratings into the calculations, which leads to a slightly higher predictive accuracy than the basic scheme.
- *Item-based Pearson*. This scheme is a traditional item-to-item CF algorithm [17] using Pearson's correlation coefficient as a similarity metric. All neighbors are taken into account for the prediction.
- *Funk-SVD*. Over the last years, matrix factorization techniques have shown to be a good basis to develop highly-accurate recommender systems. In these approaches, the goal is to automatically identify a set of latent semantic features (aspects or factors) which characterize the available items using for example Singular Value Decomposition (SVD) or probabilistic approaches. Predictions can then be made by determining and combining the information about the position of each item and user interest profile in this reduced feature space. In our evaluation, we used an algorithm recently proposed in the context of the Netflix Prize by Simon Funk (pen name)[6].
- *Factorized Neighborhood Model (Factorized NB)*. In order to combine the advantages of traditional neighborhood-based models and latent factor models, Koren very recently in [12] proposes a new method, which lead to highly accurate results on the Netflix data set. We use the item-based variant of Koren's models in our work.

We ran additional experiments to determine suitable parameters for the individual algorithms.

- In the Funk-SVD method, using 30 latent factors led to the best results.
- The parameters for the Factorized Neighborhood Model were: $\#iterations = 20$, $\#features = 200$, $\gamma = 0.001$, $\lambda = 0.002$.
- Finally, for Parameterized RF-Rec we determined the following optimal setting for the meta-parameters: $\#iterations = 15$, $\gamma = 0.001$, $\lambda = 0.002$ for all data sets except for the 100k-MovieLens data where we increased λ to 0.02.

C. Evaluation metrics and procedure

To measure the predictive accuracy of our methods we use the *Root Mean Squared Error* (RMSE), which is a standard metric in the RS research. The RMSE metric measures the average absolute deviation between the predicted rating by a recommender and a user's real (but withhold) rating. By design, it puts more emphasis on larger prediction errors. Formally,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} |\hat{r}_i - r_i|^2}{N}}$$

[4]http://www.grouplens.org/node/73

[5]http://webscope.sandbox.yahoo.com

[6]http://sifter.org/~simon/journal/20061211.html

where N stands for the total number of predictions. Lower RMSE values obviously correspond to more accurate predictions.

Beside accuracy, we also measure the *prediction coverage* of the recommenders, that is, the percentage of user-item combinations a recommender can make predictions [16]. Prediction coverage is defined as

$$COV = \sum_{i=1}^{N} \frac{\mathbf{1}(\hat{r}_i)}{N}$$

where N is the number of tested user-item combinations and the indicator function returns 1 if a prediction can be computed and 0 otherwise.

For the evaluations, we divided the rating databases into training sets and test sets. As usual, the goal is to predict the withheld ratings in the test set. In order to analyze the impact of different data density levels, we varied the density levels by using subsamples of different sizes as was also done for example in [17]. The smallest subsample used in the experiments contained 10% of the original data. In this subsample, the average number of ratings per user was around 10 and 17 for the MovieLens 100k and 1M data sets, respectively, and 3 for the Yahoo!Movies data set. Further measurements were taken in steps of 10% up to the 90% data set, which corresponds to the usual 90% train/test ratio for accuracy measurements in literature. The experiments were appropriately repeated in order to factor out randomness effects.

IV. Evaluation results

In this section we will present the results of the experimental evaluation on the three data sets. We first compare the RF-Rec prediction schemes presented above among each other and analyze the accuracy of their user- and item-based components individually. Afterwards, we compare the RF-Rec schemes with other algorithms and provide accuracy and coverage results for the different data sets. We finally compare the computational complexity of the algorithms by analyzing both the time needed for model-building, which is required for model-based approaches such as Funk-SVD when new data is available, and the prediction time needed for estimating *one* rating $\hat{r}_{u,i}$ for a given user u and item i.

A. Accuracy results

All RF-Rec prediction schemes presented above have two independent components, a user- and an item-component. In a first step, we therefore aim to analyze the contribution each component makes and measure their performance separately. Figure 1 shows a comparison of the RF-Rec predictors and their components on the 100k-MovieLens data set. Note that we do not report the results for the other data sets as the observations were similar across all data sets.

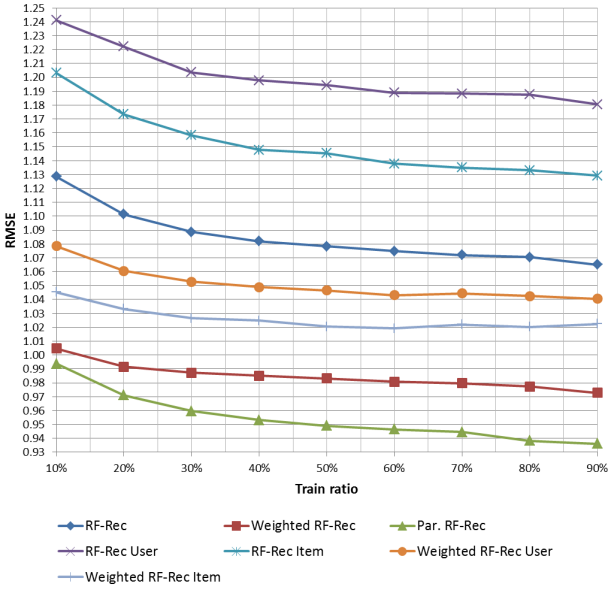

Figure 1. Comparison of the RF-Rec predictors on the 100k-MovieLens data set.

In Figure 1, the user- and item-components of the basic RF-Rec scheme are denoted as RF-Rec User and Item respectively. They basically use the most frequent rating value in the user's or the item's rating vector as the prediction. On the other hand, Weighted RF-Rec User and Item stand for the user- and item-model of the Weighted RF-Rec scheme respectively (see Equation (4)).

The results in Figure 1 show that the hybrid methods, which combine the user and the item components, always perform better than their components alone. We can also observe that the item components (RF-Rec Item and Weighted RF-Rec Item) achieve higher accuracy than their user-based counterparts, i.e., the item-based frequencies are of higher value than the user-based frequencies. Finally, we can see that the Parameterized RF-Rec scheme performed best, followed by Weighted RF-Rec. The basic RF-Rec scheme shows comparably poor performance when using the RMSE metric because RF-Rec always returns only integer values. Note that in [14] and [18], the MAE metric was used to evaluate the basic RF-Rec scheme. The evaluation on this measure demonstrated that the basic scheme leads to good predictive accuracy when compared with traditional nearest-neighbor CF algorithms.

Next, we report the results from the comparison of the RF-Rec schemes with the other algorithms. Figure 2 shows the RMSE values for different training set sizes for the MovieLens data sets.

On the 100k-MovieLens data set (Figure 2 (a)), Parameterized RF-Rec showed very good performance and consistently outperforms even the recent SVD-based algorithm (Funk-SVD). Even more, up to the 30% level,

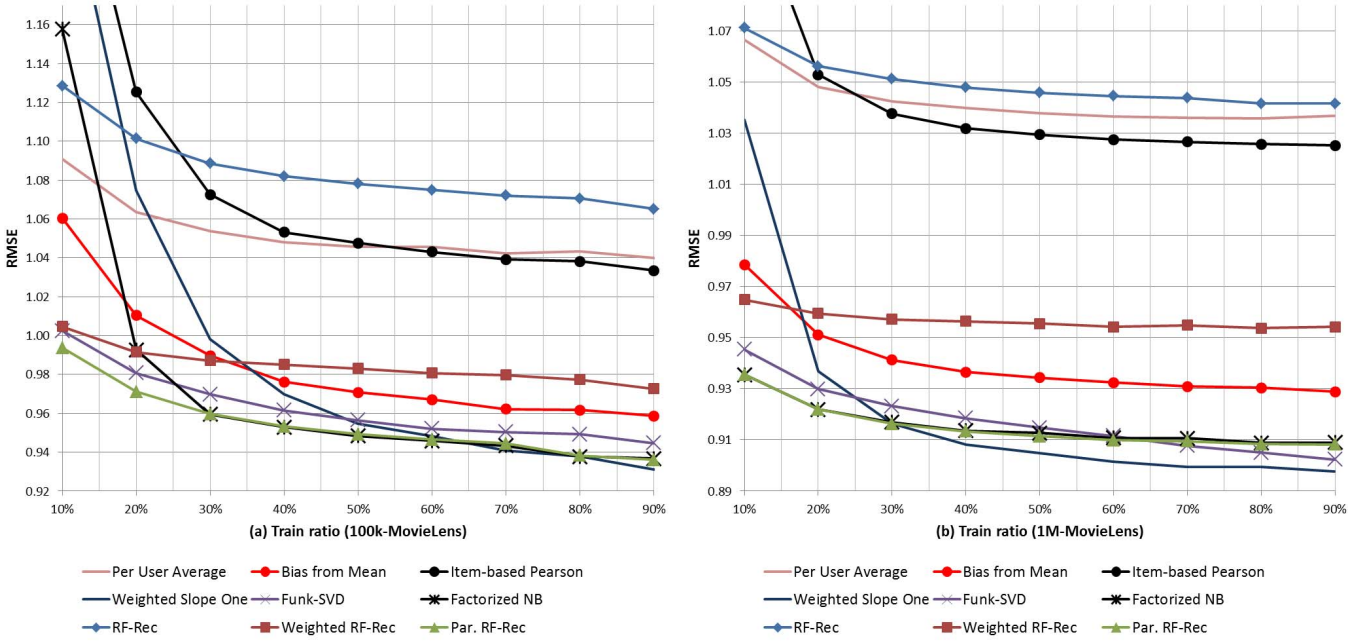

Figure 2. RMSE values for different training set sizes: 100k-MovieLens (a), 1M-MovieLens (b).

PARAMETERIZED RF-REC's recommendations are more accurate than those of all other techniques evaluated in this study. Above this level, the accuracy of PARAMETERIZED RF-REC is comparable to that of Koren's Factorized NB. Weighted Slope One also leads to high accuracy results but performs poorly in the cold start phase (up to the 40% density level). For the reasons described above the basic RF-REC scheme only achieves a relatively low prediction accuracy.

Next, we made experiments on the 1M-MovieLens data to analyze recommendation scenarios in which more rating data is available (Figure 2 (b)). The Weighted Slope One Scheme benefits from the additional data at density levels higher than 30% and achieves better accuracy results than PARAMETERIZED RF-REC. We should, however, keep in mind that a density level of 30% corresponds to a setting where the average number of ratings per user in the training data set is 50, which, in our opinion, does not fully reflect real-world scenarios in which a large share of users only rate a small number of items. In the important cold start phase, Weighted Slope One, however, again performs relatively poorly. Notably, PARAMETERIZED RF-REC performs equally well compared to Factorized NB and leads to even better accuracy results than the recent SVD-based approach up to the 70% level. In addition, note that on this level 116 ratings per user are available on average.

Figure 3 finally shows the results obtained for the Yahoo!Movies data set. On this data set the performance of the Weighted Slope One scheme deteriorates strongly such that even Bias from Mean (Non Personalized) achieves higher accuracy values. We think that the high sparsity value of the Yahoo! data set is the main reason for the poor performance of Weighted Slope One. On the other hand, Funk-SVD, Factorized NB and also PARAMETERIZED RF-REC perform well on this data set and show constantly good performance more or less on all evaluated density levels.

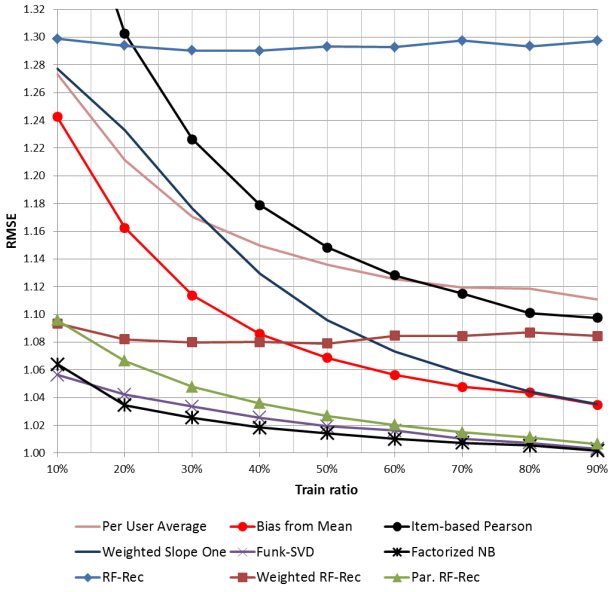

Figure 3. RMSE values for the Yahoo!Movies data set.

Table III
PREDICTION COVERAGE OF RECOMMENDATION APPROACHES (YAHOO!MOVIES).

Density level	*RF-Rec predictors*	*Item-based Pearson*	*Weighted Slope One*	*Funk-SVD*	*Factorized NB*
10%	100%	27%	89%	90%	90%
20%	100%	57%	93%	93%	93%
30%	100%	73%	95%	95%	95%
..	..	..	..	..	..
90%	100%	92%	97%	97%	97%

B. Coverage

Regarding prediction coverage, i.e., the question for which percentage of the user-item combinations a recommender can make predictions, note that in contrast to nearest-neighbor approaches that often use similarity and neighborhood size thresholds, our recommendation scheme can make predictions if at least one rating for the target item or one rating by the user is available. Table III shows the coverage of the algorithms on the sparse Yahoo data set. We can see that due to their nature the RF-REC predictors (RF-REC, WEIGHTED RF-REC and PARAMETERIZED RF-REC) lead to a 100% prediction coverage also for very sparse data sets such as the Yahoo!Movies data set. In contrast, the coverage of the Item-based Pearson method, for example, only slowly increases from 27% to 92% when increasing the density level from 10% to 90%. On this data set, the RF-REC predictors also outperform the more complex SVD and Factorized NB algorithms on the coverage metric, even though they have a good prediction coverage even on sparse data sets.

C. Computational complexity

One of the most advantageous characteristics of the RF-REC schemes is their very low computational complexity. "Model-building" for these methods mainly means to scan the user-item matrix once, count the frequencies of the rating values for each item and user and store these numbers. Also the memory requirements are very limited given the simple nature of the algorithm. Even if we, for example, have 1 million items, 1 million users and 5 rating levels, we need $2 \times 5 \times 1{,}000{,}000$ integers to store the rating frequencies, which only requires 40MB in main memory if we assume 4-byte integers as in Java. Generating predictions at run-time is also efficient with RF-REC, as we only need to look up the frequencies for the given user-item pair and do some simple (weighting) arithmetic to compute the prediction.

In the Item-based Pearson scheme, the similarities between items have to be calculated in an offline phase, see [6] for a discussion of a relatively efficient implementation. In the SVD approach, the latent user and item vectors have to be learned from the rating data in the model-building phase. Details of the learning and model-building phase of Koren's factorized neighborhood models are given in [12].

Table IV shows the model-building and prediction times of the different algorithms in milliseconds (ms). The numbers represent averages and were obtained from multiple experiments on the 1M-MovieLens data set at a density level of 90%. Note that the chosen data set is the largest one of our evaluation. The experiments were conducted on a typical desktop PC[7].

Table IV
MODEL-BUILDING AND PREDICTION TIMES IN MS (1M-MOVIELENS).

Algorithm	*Model-building time*	*Prediction time*
PARAMETERIZED RF-REC	555	0.001849
Item-based Pearson	367,040	0.107278
Weighted Slope One	39,227	0.205828
Funk-SVD	110,755	0.002647
Factorized NB	86,293	0.343836

Model-building takes 555 ms in our approach as opposed to almost 2 minutes in Funk-SVD. As a side note, also the determination of the parameters for PARAMETERIZED RF-REC using the gradient descent method is relatively quick and needs only a few seconds (2,611 ms). The gradient descent solver iterates over the ratings of each user and modifies the parameters. Therefore the overall time complexity for the training phase is $O(\sum_u |R(u)|)$ where $R(u)$ describes all ratings of a given user u. Note again, that it is not necessary to change the weight parameters upon the arrival of new data. Instead, only the lists which store the rating frequencies have to be updated (incremented).

Table IV further shows that our approach requires the least time for generating a prediction. Again there is multiple order of magnitude difference between the recommenders. For example, our approach is able to compute 540 predictions in 1 ms whereas the slowest one (Factorized NB) makes only 3 predictions in 1 ms. The prediction time is a very important factor in high-traffic e-commerce recommender systems in which up to 1,000 parallel requests have to be served [5].

V. SUMMARY AND OUTLOOK

RF-REC is a highly scalable collaborative filtering algorithm, which is easy to implement, can generate predictions in constant time, does not require a computationally intensive offline model-building phase, and at the same time has a high prediction coverage. In this paper we proposed two accuracy-improving extensions for the recent RF-REC frequency-based recommendation scheme (WEIGHTED RF-REC and PARAMETERIZED RF-REC).

[7]Intel Core2Duo T8300, 2.40GHz, 3GB Ram

An evaluation on three commonly used data sets demonstrated that comparably good accuracy results can be achieved with the proposed extensions of the basic RF-REC scheme, even when compared with more complex and recent methods based on matrix factorization. Even though some of the more sophisticated methods achieve slightly higher accuracy in some settings, they require a considerably higher model-building time to achieve a tiny improvement on the RMSE measure. Overall, our methods manage to preserve the key advantages of the basic scheme, such as constantly high prediction coverage and short model-building phase, and achieve comparable accuracy results.

Our future work includes the evaluation of the approach on additional data sets. In particular we plan to run experiments on the Netflix data in order to compare the RF-REC predictors with other approaches, which are evaluated in this data set. Since putting too much focus on accuracy metrics alone may even negatively impact the systems as reported by [19], we want to move beyond accuracy in our future work by evaluating our methods on other quality dimensions such as diversity and novelty.

Finally, by making the software used in our experiments publicly available[8], we also hope to contribute to the comparability of different RS algorithms.

REFERENCES

[1] R. Nikolaeva and S. Sriram, "The moderating role of consumer and product characteristics on the value of customized on-line recommendations," *Int. J. Electron. Commerce*, vol. 11, pp. 101–123, December 2006.

[2] M. B. Dias, D. Locher, M. Li, W. El-Deredy, and P. J. Lisboa, "The value of personalised recommender systems to e-business: A case study," in *Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys'08)*, Lausanne, Switzerland, 2008, pp. 291–294.

[3] D. M. Fleder and K. Hosanagar, "Recommender systems and their impact on sales diversity," in *Proceedings of the 8th ACM Conference on Electronic Commerce (EC'07)*, San Diego, California, USA, 2007, pp. 192–199.

[4] O. J. Daramola, M. O. Adigun, C. K. Ayo, and O. O. Olugbara, "Improving the dependability of destination recommendations using information on social aspects," *Tourismos: An International Multidisciplinary Journal of Tourism*, vol. 5, pp. 13–34, April 2010.

[5] D. Jannach and K. Hegelich, "A case study on the effectiveness of recommendations in the mobile internet," in *Proceedings of the 2009 ACM Conference on Recommender Systems (RecSys'09)*, New York, NY, USA, 2009, pp. 41–50.

[6] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, pp. 76–80, January 2003.

[8] http://ls13-www.cs.uni-dortmund.de/rec_suite.zip

[7] U. Hanani, B. Shapira, and P. Shoval, "Information filtering: Overview of issues, research and systems," *User Modeling and User-Adapted Interaction*, vol. 11, pp. 203–259, August 2001.

[8] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, June 2005.

[9] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems - An Introduction*. Cambridge University Press, 2010.

[10] M. Balabanovi and Y. Shoham, "Fab: Content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, pp. 66–72, March 1997.

[11] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI'02)*, Edmonton, Alberta, Canada, 2002, pp. 187–192.

[12] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Transactions on Knowledge Discovery from Data*, vol. 4, pp. 1:1–1:24, January 2010.

[13] N. D. Lawrence and R. Urtasun, "Non-linear matrix factorization with gaussian processes," in *Proceedings of the 26th Annual International Conference on Machine Learning (ICML'09)*, Montreal, Quebec, Canada, 2009, pp. 601–608.

[14] F. Gedikli and D. Jannach, "Recommending based on rating frequencies," in *Proceedings of the 2010 ACM Conference on Recommender Systems (RecSys'10)*, Barcelona, Spain, 2010, pp. 233–236.

[15] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," in *Proceedings of the 5th SIAM International Conference on Data Mining*, Newport Beach, CA, 2005, pp. 471–480.

[16] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, pp. 5–53, January 2004.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web (WWW'01)*, Hong Kong, 2001, pp. 285–295.

[18] F. Gedikli and D. Jannach, "Recommending based on rating frequencies: Accurate enough?" in *Proceedings of the 8th Workshop on Intelligent Techniques for Web Personalization & Recommender Systems at UMAP'10 (ITWP'10)*, Big Island, Hawaii, 2010, pp. 65–70.

[19] S. M. McNee, J. Riedl, and J. A. Konstan, "Being accurate is not enough: How accuracy metrics have hurt recommender systems," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, Montreal, Quebec, Canada, 2006, pp. 1097–1101.