Recommending based on rating frequencies: Accurate enough?

Fatih Gedikli and Dietmar Jannach

Technische Universität Dortmund,
44227 Dortmund, Germany
{firstname.lastname}@tu-dortmund.de

Abstract. Since the development of the comparably simple neighborhood-based methods in the 1990s, a plethora of techniques has been developed to improve various aspects of collaborative filtering recommender systems such as predictive accuracy, scalability to large problem instances or the capability to deal with sparse data sets. Many of the recent algorithms rely on sophisticated methods which are based, for instance, on matrix factorization techniques or advanced probabilistic models or require computationally intensive model-building phases. In this work we evaluate the accuracy of a new and extremely simple prediction method that uses the user's and the item's most frequent rating value to make a rating prediction. The evaluation on two standard test data sets shows that the accuracy of the algorithm is on a par with the standard collaborative filtering algorithms on dense data sets and outperforms them on sparse rating databases. Besides that, the algorithm's implementation is trivial, has a high prediction coverage, requires no complex offline pre-processing or model-building phase and can generate predictions in a constant time.

1 Introduction

Collaborative filtering is one of the most successful technologies for recommender systems [AT05]. Pure collaborative filtering recommender systems only rely on a given user-item rating matrix to make rating predictions for items that the *active user* has not seen yet. Early neighborhood-based recommendation schemes simply used the k nearest neighbors (kNN) as predictors for unseen items. Later on, a broad range of more advanced and sophisticated methods have been applied to better exploit the given rating information and to improve the recommendation process in one or the other dimension. Examples for such methods include matrix factorization, various probabilistic models, clustering techniques, graph-based approaches as well as machine learning techniques, based on, e.g., association rule mining, see also [AT05].

Typically, the more elaborate approaches outperform the commonly-used kNN baseline method in terms of accuracy in particular for sparse data sets or in terms of scalability as they rely on offline pre-processing or model-building phases. In [LM05], Lemire and Maclachlan formulate additional desirable features of a recommendation scheme such as that they are easy to implement, can

be updated on the fly, are efficient at query time and are "reasonably" accurate. Their evaluation shows that the proposed *Slope One* family of item-based recommender algorithms, which is based on the computation of "popularity differentials between items for users", leads despite its simplicity to relatively accurate predictions (measured in terms of Mean Absolute Error). Due to its simplicity, different implementations of the algorithm in various programming languages and frameworks are available today.

In this paper, we propose an even simpler recommendation scheme, RF-Rec, which is only based on the absolute frequencies of the different rating values per user and per item. The method is therefore trivial to implement, can generate predictions in constant time, does not require a computationally intensive offline model-building phase, and at the same time leads to competitive prediction coverage and accuracy results in particular for sparse data sets.

In the rest of the paper, we will first describe the RF-Rec recommendation scheme in more detail and present results of an experimental evaluation on two commonly-used data sets.

2 Recommending based on rating frequencies

Let us illustrate the RF-Rec recommendation scheme with a simplified and relatively sparse rating database shown in Figure 1. The goal in our example is to predict Alice's rating for item $I3$.

	I1	I2	I3	I4	I5	Average
Alice	1	1	**?**	5	4	2.75
U1	2		5	5	5	4.25
U2			1	1		1.00
U3		5	1	1	2	2.25
Average	1.50	3.00	2.33	3.00	3.67	

Fig. 1. Example user-item rating matrix.

When adopting a user-based kNN scheme, probably no prediction can be made because only one relatively similar user $U1$ exists which could be taken as a predictor for Alice. If we allow also such small neighborhood sizes, the prediction for Alice will usually consist of taking the neighbor's rating for $I3$ and using it for the prediction by making a weighted addition to Alice's average rating. Similarly, in an item-based kNN approach, Alice's rating value for item $I4$, whose rating vector is similar to the one of $I3$ will be taken as a predictor. In both cases, the prediction for Alice for item $I3$ will be rather high.

In our approach, however, the predictions are based on absolute rating frequencies. The prediction function for a given user u and an item i in the RF-Rec recommendation scheme is defined as follows:

$$pred(u,i) = \underset{r \in possibleRatings}{\arg\max} \Big(\big(freqUser(u,r) + 1 + \mathbb{1}_{avg-user}(u,r) \big) * \big(freqItem(i,r) + 1 + \mathbb{1}_{avg-item}(i,r) \big) \Big)$$

where $freqUser(u,r)$ is the frequency of ratings with value r of the target user u and $freqItem(i,r)$ is the frequency of ratings with value r of the target item i. $\mathbb{1}_{avg-user}(u,r)$ and $\mathbb{1}_{avg-item}(i,r)$ are indicator functions which return 1 if the given rating corresponds to the rounded average rating of the target user or target item accordingly and 0 otherwise.
In the example, where 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, 2:0, 3:0, 4:0, 5:1] we would do the following calculations:

Rating value 1: (2+1+0)*(2+1+0) = 9
Rating value 2: (0+1+0)*(0+1+1) = 2
...
Rating value 5: (1+1+0)*(1+1+0) = 4

Since the formula result for rating 1 is the highest, we would predict that Alice would give a "1" to item $I3$, which is strongly different from the ratings that we would predict with the other methods.

The rationale of the prediction scheme is as follows. First, instead of taking rating *averages* into account for the calculations (as done in kNN-based approaches and also in Slope One) we rely on rating *frequencies*. Intuitively, this can be advantageous in case of extreme ratings, i.e., since Alice only gave very low and very high ratings and at the same time item $I3$ also only received extreme ratings. Incorporating user or item averages would move the predictions away from these extremes. When using the Slope One scheme, 2.38 would be the predicted rating for Alice, which is slightly below her average. Note that in [HKR00], the authors *Herlocker et al.* have also observed that high variance in the rating data can lead to decreased recommendation accuracy.

The "1" in the middle of our 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 in our scheme 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, we add some small extra weight to it, thus very slightly preferring the average rating.

Regarding *prediction coverage*, that is, the question for what percentage of items a recommender can generate predictions, note that in contrast to kNN 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.

In order to measure the predictive accuracy of the method, we therefore evaluated our approach on two popular data sets using a common experimental procedure and accuracy metric. The results are described in the following section.

3 Experimental Evaluation

Algorithms, data sets and metrics. As data sets for the evaluation, we used the 100k-MovieLens rating database (100,000 ratings by 943 users on 1,682 items) and a snapshot of the Yahoo!Movies data set (211,231 ratings by 7,642 users on 11,915 items)[1]. The MovieLens data set only contains users who have rated at least 20 items; the minimum number of rated items per user in the Yahoo! data set is 10.

The density level of the data sets were varied by using subsamples of different sizes. The smallest subsample contained 10% of the original data. In this subsample, the average number of ratings per user was around 10 for the MovieLens data set 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 *Mean Absolute Error* (MAE) measurements.

We compared the following algorithms: user-based kNN (using default voting, Pearson similarity and the neighborhoodsize of 30 as suggested as optimal value in literature), item-based kNN (Mahout's item-based kNN-method implementation with Pearson similarity)[2], Slope One [LM05], Bias from Mean, Per User Average, the recent recursive prediction algorithm (RPA) [ZP07] (using larger, empirically determined neighborhood sizes of 100) and RF-Rec.

Results. Figure 2 (a) shows the MAE values for different training set sizes for the MovieLens data set. Up to the 50% level, RF-Rec has consistently better accuracy than all other techniques. Above that level, the accuracy of RF-Rec (0.742) is comparable to Slope One (0.743) and RPA (0.734). Note that RF-Rec due to its nature leads to 100% prediction coverage also for very sparse data sets. In contrast, the coverage of the user-based kNN method, for example, slowly increases from 60% to 95% when increasing training set ratio from 10% to 90%, see Figure 3. In the experiments, we further varied the neighborhood size ns of the user-based kNN method. Increasing ns to 300 lead to the observation that the accuracy improved and was comparable to the one of RPA for training set sizes higher than 50%.

Figure 2 (b) shows the results for the Yahoo! data set. We can observe similar accuracy values also for this data set. In particular, the improvement of our RF-Rec algorithm is even stronger on that data set. A possible explanation for this observation could be the different sparsity levels of the two data sets, i.e., assuming that RF-Rec works particularly well for sparse settings, it is intuitive that even better results can be achieved on the sparser Yahoo! data set (0.9976 sparsity) than on the MovieLens data set (0.9369 sparsity).

[1] http://www.grouplens.org/node/73, http://webscope.sandbox.yahoo.com
[2] http://lucene.apache.org/mahout

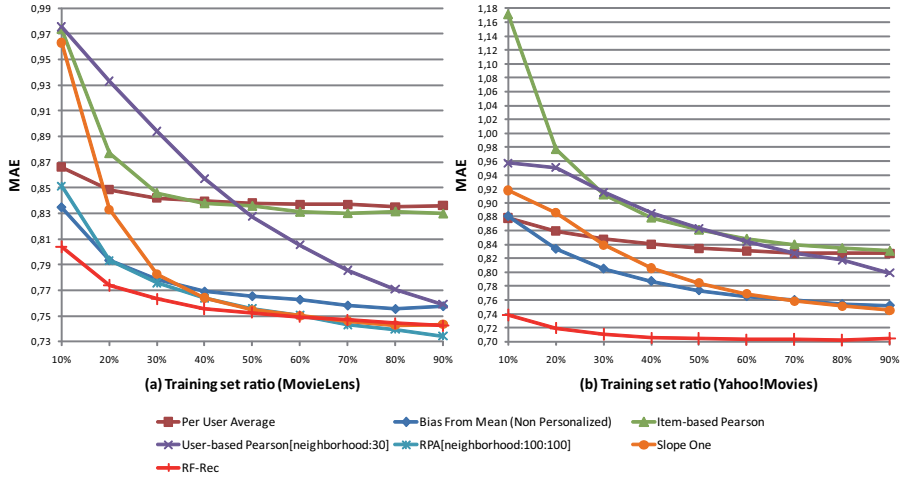

Fig. 2. MAE values for different training set sizes: MovieLens (a), Yahoo!Movies (b).

Training set ratio	RF-Rec	Slope One	User-based kNN	Item-based kNN
10%	100%	97%	60%	44%
20%	100%	98%	58%	94%
30%	100%	99%	70%	98%
..	..	..	..	..
90%	100%	99%	95%	99%

Fig. 3. Prediction coverage of recommendation approaches (MovieLens).

Overall, these accuracy findings indicate that RF-Rec has a constantly good performance which is quite independent of the training set ratio. RF-Rec is despite its simplicity suitable to generate predictions with an accuracy which is comparable to existing approaches and is even better for sparse data sets, which can often be found in practice.

Computational complexity. In the RF-Rec scheme, the "model-building" phase obviously consists of calculating the frequencies of the individual rating values per user and per item, which can be accomplished in a single scan of the matrix; the frequency statistics can be easily updated when new ratings are available. Given u users, i items and v possible rating values, the memory requirements for the model are constant: $(u * v) + (i * v)$. Also the calculation of predictions can be done with the formula from Section 2 in constant time. In absolute numbers, "model-building" requires less than 10 seconds even when hundreds of millions of ratings exist; predictions can be calculated in a few milliseconds on a standard desktop computer. We compared our method with Mahout's item-based kNN-method implementation on the 1 million MovieLens data set: model-building takes 500ms in our approach as opposed to 6 minutes with

Mahout. Generating a prediction takes only 3ms in our framework (and 100ms with Mahout), which is a very important factor in high-traffic recommenders in which up to 1,000 parallel requests have to be served [JH09].

4 Summary

In this work we proposed a new, frequency-based recommendation scheme that leads to good predictive accuracy and is at the same time highly scalable and very easy to implement. Our future work includes the evaluation of the approach on the Netflix data in order to compare it to the results of more recent methods; in addition, we will also compare the predictive accuracy of the different methods based on precision and recall.

Overall, our evaluation demonstrated that comparably good results can be achieved with simple methods and that the accuracy values that can be achieved with the help of "classical" methods such as item-based kNN and even the more recent RPA method are actually very small, which could make the payoff of using more sophisticated methods in some settings questionable.

We hope that light-weight approaches like our RF-Rec method help to further promote the use of recommender systems in practice; by making the software used in our experiments publicly available[3], we hope to contribute to the comparability of different algorithms since our study revealed that relevant algorithmic details and parameters are often not reported in sufficient detail.

References

[AT05] Gediminas Adomavicius and Alexander Tuzhilin, *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions*, IEEE Trans. on Knowl. and Data Eng. **17** (2005), no. 6, 734–749.

[HKR00] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl, *Explaining collaborative filtering recommendations*, Proc. ACM Conference on Computer Supported Cooperative Work (New York, NY, USA), 2000, pp. 241–250.

[JH09] Dietmar Jannach and Kolja Hegelich, *A case study on the effectiveness of recommendations in the mobile internet*, Proceedings of the 2009 ACM Conference on Recommender Systems (New York, NY, USA), 2009, pp. 41–50.

[LM05] Daniel Lemire and Anna Maclachlan, *Slope one predictors for online rating-based collaborative filtering*, Proceedings of the 5th SIAM International Conference on Data Mining (Newport Beach, CA), 2005, pp. 471–480.

[ZP07] Jiyong Zhang and Pearl Pu, *A recursive prediction algorithm for collaborative filtering recommender systems*, Proceedings of the 2007 ACM Conference on Recommender Systems (Minneapolis, MN, USA), 2007, pp. 57–64.

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