Improved Listwise Collaborative Filtering with High-Rating-Based Similarity and Temporary Ratings

Yoshiki Tsuchiya
University of Tsukuba
Tsukuba, Japan
tsuchiya@cmu.iit.tsukuba.ac.jp

Hajime Nobuhara
University of Tsukuba
Tsukuba, Japan
nobuhara@iit.tsukuba.ac.jp

ABSTRACT
In this paper, we make two proposals to speed up listwise collaborative filtering and improve its accuracy. The first is to speed up computation by only using a subset of the rating information (the high ratings). The second is to improve accuracy using temporary ratings that estimate the rating scores that neighboring users are not rating. Experiments using MovieLens datasets (1M and 10M) demonstrate that these proposals effectively reduce computation time about 1/50 and improve accuracy 2.22% compared with ListCF, a well-known listwise collaborative filtering algorithm.

Author Keywords
Recommender system; Ranking-oriented collaborative filtering;

ACM Classification Keywords
• Information systems, Collaborative filtering

INTRODUCTION
In recent years, due to the development of the Web, recommender systems have become increasingly important in various situations; many researchers are now focusing on recommendation technologies and systems [2,6,7,9,10]. Collaborative filtering (CF) is a widely used recommendation algorithm that is based on the similarity between users or items, as calculated from a user and rating matrix. Various CF algorithms have been proposed, and they can be divided into two types: rating-oriented [6,9] and ranking-oriented [2,7,10], as shown in Fig. 1. Rating-oriented CF algorithms, such as item-based CF [9], predict the ratings of items that have not been evaluated by users and make recommendations by calculating the similarity between users or items. On the contrary, ranking-oriented CF uses user similarity to predict the item ranking and recommends items based on this. We will focus on this method due to its performance. Ranking-oriented CF can be further divided into two types: pairwise ranking-oriented [7,10] and listwise ranking-oriented [2]. Pairwise ranking-oriented CF predicts the order of pairs of items but requires large computation time.

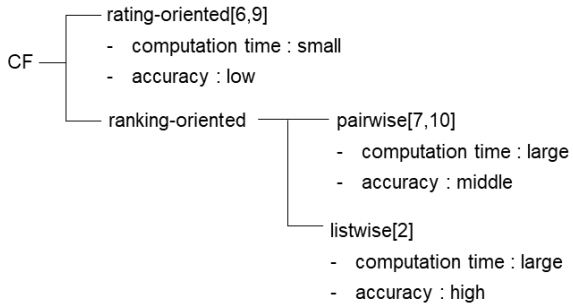

Figure 1. Collaborative filtering classification

In contrast, listwise ranking-oriented CF predicts the order of the complete list of items. Although this produces better accuracy than a typical pairwise CF algorithm, calculating the required similarities is time-consuming and there is room to improve the ranking accuracy. In this paper, we propose an efficient listwise ranking-oriented CF algorithm that is both faster and has higher ranking accuracy.

The proposed method implements two improvements. First, when calculating the similarity between users, it only considers the highest-rated items, greatly speeding up the calculation. Second, it introduces temporary ratings when making ranking predictions. Experimental comparisons using MovieLens 1M (6,040 users, 3,952 movies, 1,000,209 ratings) and 10M (71,567 users, 10,681 movies, 10,000,054 ratings) confirm that the proposed method reduces about 1/50 computation time of similarity and improves 2.22% ranking accuracy than a conventional CF algorithm.

RELATED WORK

Overview of ListCF
In this section, we give an overview of ListCF [2], a well-known ranking-oriented listwise CF algorithm. ListCF proceeds in two phases, first calculating the similarities between users and then predicting ranks for the target user's unrated items. The first phase is based on a probability distribution of item permutations, calculated by combining the Plackett–Luce [8] and top-k probability [1] models and finding each user's neighboring users.

WII'18, March 11, Tokyo, Japan.

Similarity calculation

In ListCF, the similarity of a pair of users u and v is calculated based on a probability distribution of item permutations for each user, calculated using the Plackett–Luce model [8], which is a representative permutation probability model. The flow of similarity calculation is shown on the left of Fig. 2 and Fig. 3. Let the set $I = \{i_1, i_2, \cdots, i_n\}$ of items $\pi^i = \left(\pi_1^i, \pi_2^i, \cdots, \pi_n^i\right) (\in I^n)$ be an ordered list where $\pi_j^i \in I$ and $\pi_j^i \neq \pi_k^i$ if $j \neq k$, and let Ω^I $(\subset I^n)$ be the set of all possible permutations of I. Given the item ratings $\left(r_{\pi_1^i}, r_{\pi_2^i}, \cdots, r_{\pi_n^i}\right)$ (e.g., on a real interval in the case of MovieLens [1,5]), where $r_{\pi_j^i}$ is the rating score of π_j^i, the probability of π^i, $P(\pi^i)$, is defined using an increasing and strictly positive function $\Phi(\cdot) \geq 0$ as follows:

$$P(\pi^i) = \prod_{j=1}^{n} \frac{\Phi\left(r_{\pi_j^i}\right)}{\sum_{k=j}^{n} \Phi\left(r_{\pi_k^i}\right)} \ (\in [0,1]), \tag{1}$$

where the function is defined as $\Phi(r) = e^r$. However, this requires us to consider $n!$ different permutations of the n items, which would take a long time to compute. To speed up the computation, the top-k probability model g_k [1] is introduced as follows:

$$g_k(i_1, i_2, \cdots, i_k) = \left\{\pi^l \middle| \pi^l \in \Omega^I, \pi_j^l = i_j, j = 1, \cdots, k, l = 1, \cdots, \frac{n!}{(n-k)!}\right\} \ (\subset \Omega^I), \tag{2}$$

and the probability of the top-k permutation is calculated as

$$P\left(g_k(i_1, i_2, \cdots i_k)\right) = \prod_{j=1}^{k} \frac{\Phi\left(r_{\pi_j}\right)}{\sum_{l=j}^{n} \Phi\left(r_{\pi_l}\right)} (\in [0,1]), \quad \forall j = 1, \cdots, k : \pi_j = i_j. \tag{3}$$

Let g_k^I $(\subset \Omega^I)$ be the set of top-k permutations of I, and let the probabilities of these permutations form the probability distribution. Then, define $I_{u,v}$ $(\subset I)$ as the set of items rated

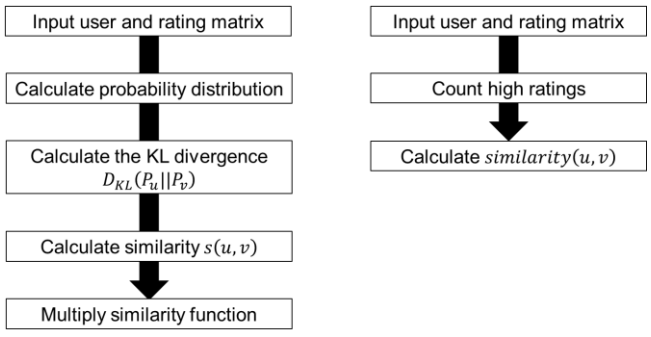

Figure 2. Similarity calculation: conventional ListCF (left), proposed method (right).

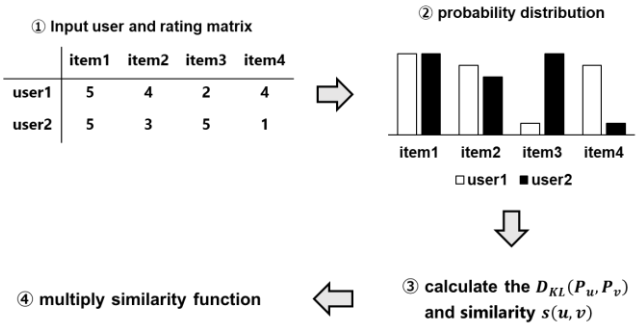

Figure 3. Procedure for similarity calculation of conventional ListCF.

by users u and v, and P_u and P_v $(\in [0,1])$ as the probability distributions over $g_k^{I_{u,v}}$ $(\subset \Omega^{I_{u,v}})$ calculated by Eq. (3) using the users' rating scores. The similarity score is now obtained from the Kullback–Leibler (KL) divergence [5] calculated from P_u and P_v. Given a pair of users u and v, the KL divergence of P_u and P_v is defined as

$$D_{KL}(P_u \parallel P_v) = \sum_{g \in g_k^{I_{u,v}}} P_u(g) \, log_2 \left(\frac{P_u(g)}{P_v(g)}\right). \tag{4}$$

The KL divergence is asymmetric, but ListCF defines the similarity $s(u,v) (\in (-\infty, 1])$ of users u and v to be symmetric as follows:

$$s(u,v) = 1 - \frac{1}{2}[D_{KL}(P_u \parallel P_v) + D_{KL}(P_v \parallel P_u)]. \tag{5}$$

If the set $I_{u,v}$ only includes a few items, the similarity will be high, so this is relaxed by multiplying by the similarity function by $\min\{I_{u,v}/c, 1\}$, where c is a threshold. Each user's neighboring users can then be found from the similarities calculated using Eqs. (3)–(5).

Ranking prediction

The flow of ranking prediction is shown on the left of Fig. 4. Let U be the set of users, $N_u (\subset U)$ be the set of the user u's neighbors and $T_u (\subset I \setminus I_u)$ be the set of items whose ranks are to be predicted. Let $\hat{P}_u$ $(\in [0,1])$ be the probability distribution of the top-k permutations $g_k^{T_u}$ $(\subset \Omega^{T_u})$ of T_u, written as

$$\hat{P}_u(g) = \frac{\varphi_{u,g}}{\sum_{g' \in g_k^{T_u}} \varphi_{u,g'}}, \tag{6}$$

where $\{\varphi_{u,g} | \forall g \in g_k^{T_u}\}$ are unknown variables assigned to the top-k permutations. In ListCF, the cross entropy is used as a loss function for prediction. Consider a target user u, for whom you want to rank a set of items T_u, and a neighboring user $v \in N_u$. Let the set of items rated by v be I_v, with $T_{u,v} = T_u \cap I_v$, and $g_k^{T_{u,v}}$ $(\subset \Omega^{T_{u,v}})$ be the set of top-k permutations of $T_{u,v}$. The cross entropy is calculated

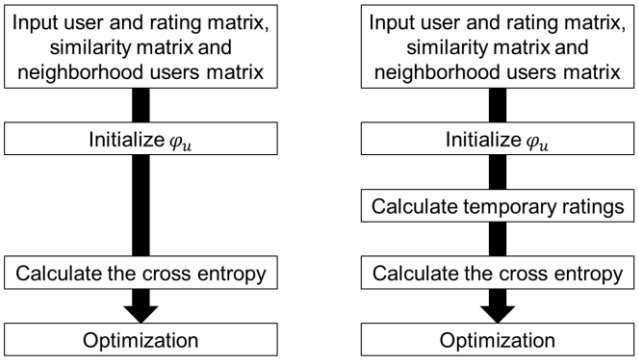

Figure 4. Ranking prediction: conventional ListCF (left), proposed method (right).

using probability distributions $\hat{P}'_u$ and P'_v over $g_k^{T_{u,v}}$ as follows:

$$E(\hat{P}'_u, P'_v) = -\sum_{g \in g_k^{T_{u,v}}} P'_v(g) \log_2 \hat{P}'_u(g). \quad (7)$$

ListCF makes predictions by minimizing the following cross entropy weighted sum:

$$\arg\min_{\varphi_u} \sum_{v \in N_u} s(u,v) \cdot E(\hat{P}'_u, P'_v), \quad (8)$$

$$\text{s.t } \forall g \in g_k^{T_u} : \varphi_{u,g} \geq 0.$$

The objective function $F(\varphi_u)$ in Eq. (8) is then transformed as follows, using Eqs. (5) and (7):

$$F(\varphi_u) = \sum_{v \in N_u} s(u,v) \cdot$$

$$\sum_{g \in g_k^{T_{u,v}}} P'_v(g) \left[\log_2 \left(\sum_{g' \in g_k^{T_{u,v}}} \varphi_{u,g'} \right) - \log_2(\varphi_{u,g}) \right] \quad (9)$$

Equation (9) is optimized by the gradient descent method. Partially differentiating F with respect to $\varphi_{u,g}$, we obtain

$$\frac{\partial F}{\partial \varphi_{u,g}} =$$

$$\sum_{v \in N_u} \frac{s(u,v)}{\ln 2 \cdot \sum_{g' \in g_k^{T_{u,v}}} \varphi_{u,g'}} - \frac{\sum_{v \in N_u} s(u,v) P'_v(g)}{\ln 2 \cdot \varphi_{u,g}}. \quad (10)$$

For a given learning rate η, $\varphi_{u,g}$ is updated as follows:

$$\varphi_{u,g} \leftarrow \varphi_{u,g} - \eta \frac{\partial F}{\partial \varphi_{u,g}}. \quad (11)$$

PROPOSED METHOD

Rapid similarity calculation using only high ratings

Conventional ListCF is slow because it has to calculate similarities using all rating scores. To speed up computation, we now propose a method of calculating user similarity by focusing only on high ratings. The flow of similarity calculation is shown on the right of Fig. 2 and Fig.5. Let H_u $(\subset I_u)$ be the set of items that were rated highly by the target user u, and $H_{u,v}$ $(\subset I_{u,v})$ be the set of items that both u and v rated highly. Given the user and rating matrix, we define the similarity between u and v as follows:

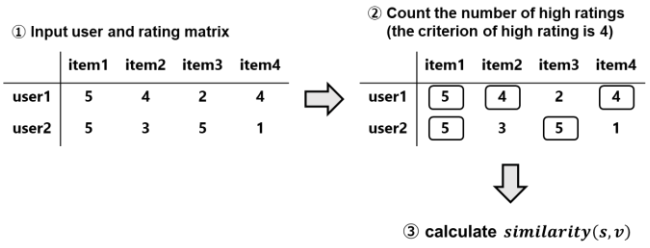

Figure 5. Procedure of similarity calculation of proposed method.

$$similarity(u,v) = \frac{|H_{u,v}|}{|H_u|} \ (\in [0,1]), \quad (12)$$

where $similarity(u,v)$ is asymmetric. We define $similarity(u,v) = 0$ if $u = v$ not to add him/herself to neighborhood users when predicting each user's ranking. In conventional ListCF, based on the top-k probability model, the time complexity is $n!/(n-k)!$, which is smallest (i.e., n) when $k = 1$. As the proposed method's time complexity is n, it is never worse than that of conventional ListCF, and often better. Changing how the high ratings are determined changes the ranking accuracy, as explained in the experimental section.

Improving ranking accuracy using temporary ratings

We also propose to improve the ListCF's ranking accuracy by introducing temporary ratings. In ListCF, the cross entropy in Eq. (7) is calculated from the probability distribution of $g_k^{T_{u,v}}$, the set of permutations of $T_{u,v}$. However, optimizing the objective function may fail if $T_{u,v}$ has too few elements, e.g., if it only includes one element and that item received a low rating from the neighboring user v. In this case, despite the item's low rating, optimization generates a high item rank, which is unhelpful for user u. Fig. 6 shows how the probability distribution for target user u's unrated items before optimization (upper) and neighboring user v's distribution (middle) give a graph like the one on the lower. Item 1 was given a low rating by v but, since no other items were rated, it is guaranteed to top the rankings. As a result, u's probability distribution is updated to place item 1 at a higher position. The proposed method therefore makes ranking predictions after giving temporary estimated ratings to items that were not rated by the neighboring user v. The flow of ranking prediction is shown on the right of Fig 4 and Fig.7. Let $r_{u,i}$ be the rating given by user u to item i, and let unrated items have a score of zero.

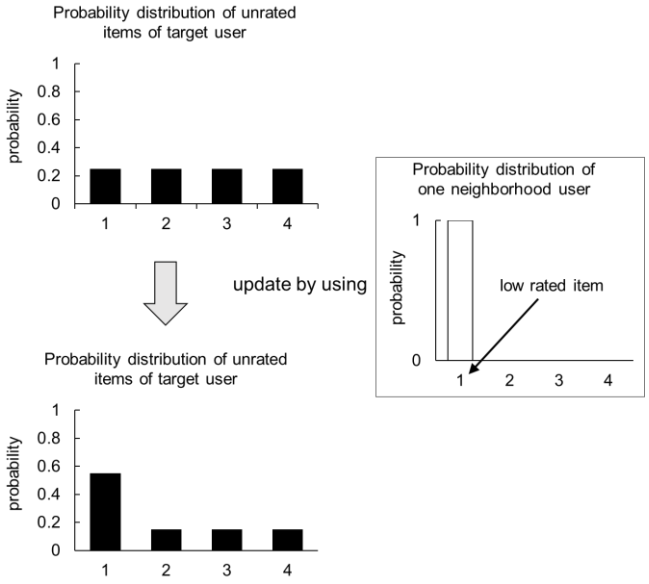

Figure 6. Examples of unsuitable parameter updates that occur when users have only one low-rated item in common. The horizontal axis represents the item number.

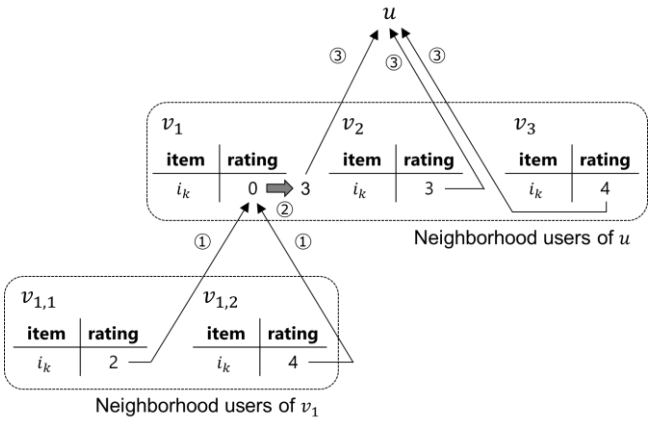

Figure 7. Procedure for calculating temporary ratings.

For a given neighborhood user $v \in N_u$ and set $T_u = \{t_1, t_2, \cdots, t_p\} (\subset I \setminus I_u)$ of items not rated by u, v's temporary ratings are defined as follows:

$$temporary\ rating(v, t_j) = \begin{cases} r_{v,t_j} & \left(\text{if } r_{v,t_j} \text{ is nonzero}\right) \\ \dfrac{\sum_{v' \in N_v} r_{v',t_j}}{\left|N'_{v,t_j}\right|} & \left(\text{if } r_{v,t_j} \text{ is zero}\right), (j = 1, \cdots, p), \end{cases} \quad (13)$$

where N'_{v,t_j} is the set of v's neighbors $v' \in N_v$ who have rated item t_j. If none of v's neighbors have rated t_j, the temporary rating will still be zero. In that case, we can obtain a nonzero temporary rating by calculating the temporary ratings $temporary\ rating(v', t_j)$ for each neighbor v' of v. By calculating the cross entropy of Eq. (7) using these temporary ratings (Eq. (13)) and replacing $s(u, v)$ in Eq. (8) with $similarity(u, v)$ (Eq. (12)), ranking predictions can then be made in the same way as for ListCF by using Eqs. (8)–(11). Calculating temporary ratings allows the neighbors' probability distributions in situations such as Fig. 6 to be more reasonable. As a result, since the parameters are less frequently updated in an undesirable way, this should improve the ranking accuracy.

EXPERIMENTAL EVALUATION

Overview of the experiment

To confirm the effectiveness of the two proposed changes, we experimentally compared the methods in terms of computation time and ranking accuracy using the MovieLens 1M and 10M datasets, details of which are shown in Table 1. We selected 10 sets of ratings from each user's rating information to create the test dataset and used the rest as training data. In this experiment, since it is shown that the ranking accuracy does not improve greatly by increasing the value of k [2], we compared the proposed method with conventional ListCF based on the top-1 probability model, as in [3,4]. Also, the threshold c of the similarity function is 300. The normalized discounted cumulative gain (NDCG) metric was used to assess ranking prediction accuracy. The NDCG value, considering the top n predicted rankings for user u, is defined as follows:

$$NDCG_u@n = Z_u \sum_{p=1}^{n} \frac{2^{r_u^p} - 1}{\log_2(1+p)}, \quad (14)$$

where Z_u is a normalization term that ensures the NDCG value of the correct ranking is 1 and r_u^p is the rating of the pth-ranked item for user u. For a set of users U, the overall NDCG@n score is calculated as follows:

$$NDCG@n = \frac{1}{|U|} \sum_{u=1}^{|U|} NDCG_u@n. \quad (15)$$

Comparison of similarity computation time

In this experiment, we measured the similarity computation time. We compared the time taken to calculate the user similarities using both the proposed and ListCF methods, and then compared the ranking accuracy of ListCF predictions made using the calculated similarities. The criterion used to determine high ratings was changed from 1 to 5 in increments of 1, and similarities were obtained for each criterion. A criterion of 1 means all ratings are used, while a criterion of 5 means that only items rated as 5 are used.

	MovieLens 1M	MovieLens 10M
Users	6,040	71,567
Items	3,952	10,681
Ratings	1,000,209	10,000,054

Table 1. Detailed Explanation of Dataset

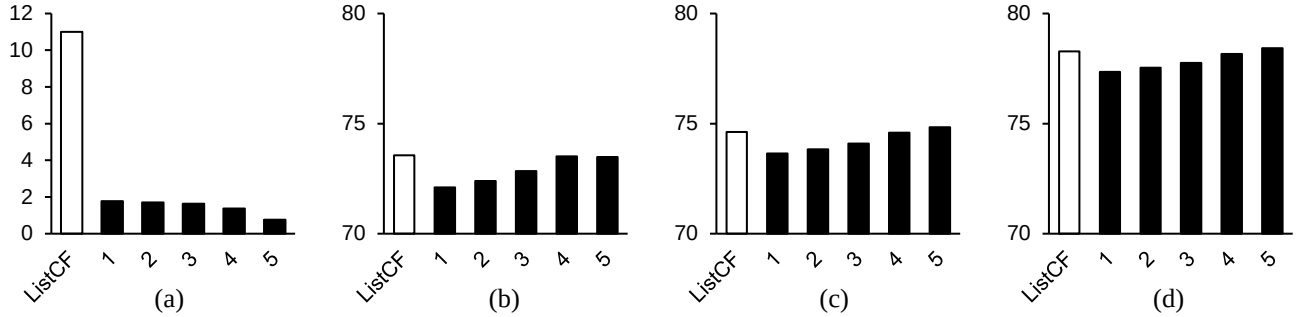

Figure 8. (a) is the comparison of calculation times of similarity and (b)-(d) are comparison of ranking accuracy by using MovieLens 1M. All horizontal axes represent criteria of high rating and each vertical axis represents minutes (a), NDCG@1 (b), NDCG@3 (c) and NDCG@5 (d) respectively.

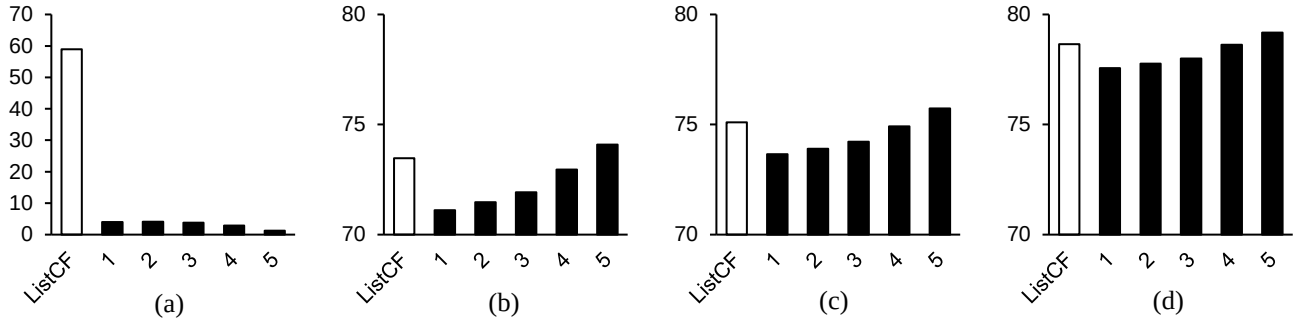

Figure 9. (a) is the comparison of calculation times of similarity and (b)-(d) are comparison of ranking accuracy by using MovieLens 10M. All horizontal axes represent criteria of high rating and each vertical axis represents hours (a), NDCG@1 (b), NDCG@3 (c) and NDCG@5 (d) respectively.

Figs. 8(a) and 9(a) show the computation time results for MovieLens 1M and 10M, respectively, while Figs. 8(b)-(d) and 9(b)-(d) show the corresponding ranking accuracy results. The horizontal axes show the high rating criterion in all graphs, while the vertical axes show calculation time in Figs. 8(a) and 9(a), NDCG@1 in Figs. 8(b) and 9(b), NDCG@3 in Figs. 8(c) and 9(c) and NDCG@5 in Figs. 8(d) and 9(d) respectively. As can be seen from Figs. 8(a) and 9(a), similarity computation is considerably more rapid for the proposed method than for conventional ListCF. In addition, Figs. 8(b)-(d) and 9(b)-(d) show that when scores of 4 and 5 are considered to be high ratings, the rankings are as accurate as those of the conventional method.

Comparison of ranking accuracy

Next, we conducted experiments to examine the effect of using temporary ratings on ranking prediction accuracy. We compared the accuracy of ranking predictions made using both the proposed and ListCF methods using the similarities calculated in the previous subsection. We used an initial $\varphi_{u,g}$ value of 10 for both datasets with learning rates of 0.025 and 0.01 for MovieLens 1M and 10M, respectively. The gradient descent method was repeated 50 times. The computation time results are shown in Fig. 10, while ranking accuracy results are shown in Figs. 11 and 12.

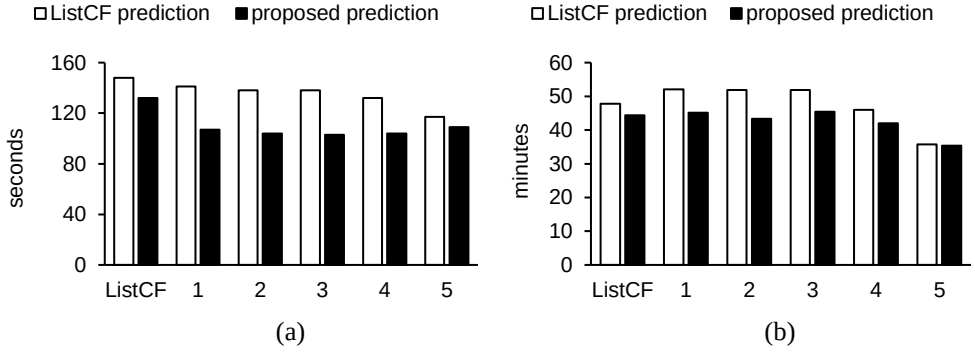

Figure 10. Ranking prediction times for MovieLens 1M (a) and MovieLens 10M (b).

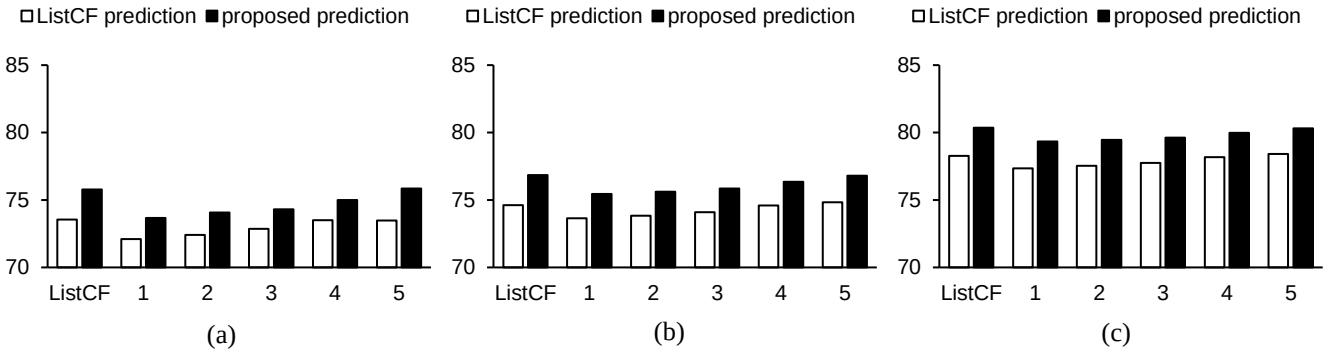

Figure 11. Ranking accuracy for MovieLens 1M. Each vertical axis represents NDCG@1 (a), NDCG@3 (b) and NDCG@5 (c).

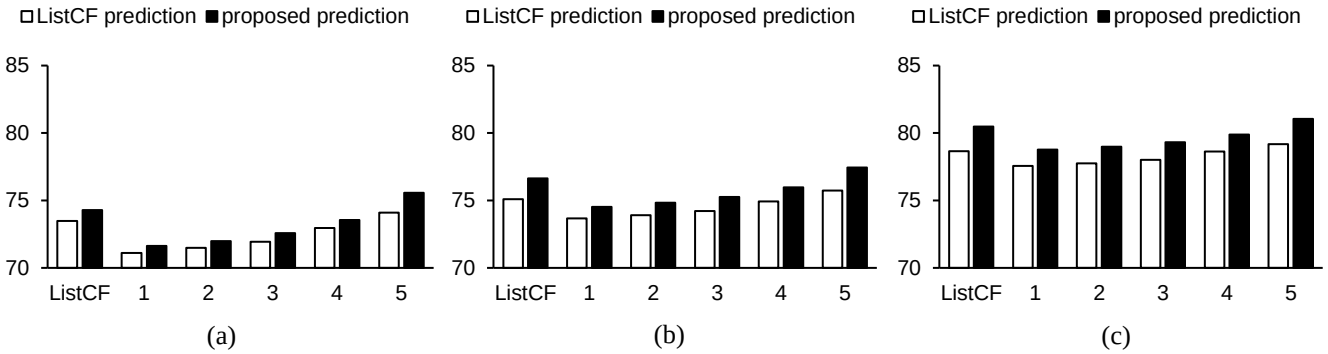

Figure 12. Ranking accuracy for MovieLens 10M. Each vertical axis represents NDCG@1 (a), NDCG@3 (b) and NDCG@5 (c).

Fig. 10 shows that the computation times for the proposed ranking prediction method were shorter in all cases. In addition, Figs. 11 and 12 show that its NDCG@1, NDCG@3 and NDCG@5 values were better for all cases. Since MovieLens 1M has less data than MovieLens 10M, the number of temporary ratings increases, and the difference between the proposed method and the conventional ListCF becomes larger than MovieLens 10.

CONCLUSION

In this paper, we have made two proposals. The first is to calculate user similarity scores using only high ratings, instead of all ratings, to speed up computation. The second is to introduce temporary estimated ratings for items that have not been rated by neighboring users to improve ranking accuracy.

In experiments using the MovieLens 1M and 10M datasets, we have compared the computation time and accuracy of both proposals with those of conventional ListCF. The results demonstrated that the first proposal provided a considerable reduction in computation time, compared with conventional ListCF, while maintaining equal or greater ranking accuracy. In addition, the second proposal shortened the prediction time in most cases while always improving the ranking accuracy.

In the future, we plan on improving the way neighboring users are selected and search for a better objective function.

REFERENCES

1. Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning* (ICML '07), 129-136.

   http://dx.doi.org/10.1145/1273496.1273513

2. Shanshan Huang, Shuaiqiang Wang, Tie-Yan Liu, Jun Ma, Zhumin Chen, and Jari Veijalainen. 2015. Listwise Collaborative Filtering. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '15), 343-352.
   http://dx.doi.org/10.1145/2766462.2767693

3. Kalervo Järvelin and Jaana Kekäläinen. 2000. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval* (SIGIR '00), 41-48.
   http://dx.doi.org/10.1145/345508.345545

4. Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. ACM Trans. *Inf. Syst.* 20, 4: 422-446.

5. S.Kullback. 1997. Information Theory and Statistics. Courier Corporation.

6. Linden, G., Smith, B., & York, J. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7, 1: 76-80.

7. Nathan N. Liu and Qiang Yang. 2008. EigenRank: a ranking-oriented approach to collaborative filtering. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (SIGIR '08), 83-90. http://dx.doi.org/10.1145/1390334.1390351

8. J. I. Marden. 1996. Analyzing and modeling rank data. CRC Press.

9. Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (WWW '01), 285-295. http://dx.doi.org/10.1145/371920.372071

10. Shuaiqiang Wang, Jiankai Sun, Byron J. Gao, and Jun Ma. 2014. VSRank: A Novel Framework for Ranking-Based Collaborative Filtering. ACM Trans. *Intell. Syst. Technol.* 5, 3: 24.