Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 6562371, 9 pages
https://doi.org/10.1155/2017/6562371

Research Article

Normalizing Item-Based Collaborative Filter Using Context-Aware Scaled Baseline Predictor

Wenming Ma,[1] Junfeng Shi,[2] and Ruidong Zhao[2]

[1] *School of Computer and Control Engineering, Yantai University, Yantai 264005, China*
[2] *Research Institute of Petroleum Exploration and Development, PetroChina, Beijing 100083, China*

Correspondence should be addressed to Wenming Ma; mwmytu@126.com

Received 5 November 2016; Revised 26 January 2017; Accepted 9 March 2017; Published 5 April 2017

Academic Editor: Bogdan Dumitrescu

Item-based collaborative filter algorithms play an important role in modern commercial recommendation systems (RSs). To improve the recommendation performance, normalization is always used as a basic component for the predictor models. Among a lot of normalizing methods, subtracting the baseline predictor (BLP) is the most popular one. However, the BLP uses a statistical constant without considering the context. We found that slightly scaling the different components of the BLP separately could dramatically improve the performance. This paper proposed some normalization methods based on the scaled baseline predictors according to different context information. The experimental results show that using context-aware scaled baseline predictor for normalization indeed gets better recommendation performance, including RMSE, MAE, precision, recall, and nDCG.

1. Introduction

The abundance of information available on the Internet makes the increasing difficulty in finding what the people want, especially for the Electronic Commerce domain. As a consequence, building personalized information selection models is becoming crucial. Among many different information selection technologies, the recommendation systems are greatly developed due to their application on most of the famous online shopping companies [1, 2].

The algorithms of recommending items have been studied extensively, most of which belong to two main categories. Content-based recommendation systems try to recommend items according to the users' past preference [3–5], whereas the collaborative recommendation systems make the recommendation in terms of the similar neighborhood preference [6–9]. Recommendation systems based purely on content generally easily suffer from the problems of limited content analysis and overspecialization. Defining the appropriate items' features is very difficult for many situations, and these features depend heavily on the users' history, which cannot find the latent profiles for recommendation.

Collaborative filter (CF) approaches overcome some of the limitations of content-based ones. Items for which the content is not available or difficult to obtain can still be recommended to users through the feedback of other users. CF ones can also recommend items with very different content, as long as other users have already shown interest for these different items. Among collaborative recommendation approaches, methods based on nearest neighbors still enjoy a huge amount of popularity, due to their simplicity, their efficiency, and their ability to produce accurate and personalized recommendations [10–12]. CF models try to capture the interactions between users and items that produce the different rating values. However, many of the observed rating values are due to effects associated with either users or items, independently of their interaction. A principal example is that typical CF data exhibit large user and item biases, that is, systematic tendencies for some users to give higher ratings than others and for some items to receive higher ratings than others.

Item-based collaborative filter [13, 14] has much more accuracy than user-based one [15, 16], when the number of items is larger than the number of users. The electronic commercial business always has huge productions. The number of productions far exceeds the number of users. However, the average number of common ratings is very small, because most of the users only have interest in very few items. User-based collaborative filter systems easily suffer from overfitting problems in this situation. So the item-based collaborative filter algorithms play an important role in modern commercial recommendation systems (RSs). This paper intends to improve the recommendation performance using a novel rating normalization strategy.

When it comes to assigning a rating to an item, each user has its own personal scale. Even if an explicit definition of each of the possible rating is supplied, some users might be reluctant to give high/low scores to items they liked/disliked. There are some different rating normalization schemes which are designed for different reasons [17–19]. Also, many of the observed rating values are due to effects associated with either users or items, independently of their interaction. We do not only convert individual ratings to a more universal scale but also consider the user and item biases.

The baseline predictor (BLP), which combines the overall averaging rating and user or item biases, involves these factors for normalization. But, for the item-based collaborative filter systems, the BLP is always a statistical constant which cannot be adaptively changed according to the context [20–23]. We found that the recommendation performance can be improved if we slightly scale the different parts of the BLP in a limited range. In this paper, we provided some novel context-aware scaled baseline predictors (CASBLP) for item-based collaborative filter normalization, considering different context information. The experimental results show that CASBLP can significantly improve the prediction performance, such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), precision, recall, and Normalized Cumulative Discounted Gain (nDCG).

The rest of this paper is organized as follows. We present the details of CASBLP in Section 2 and show experimental results in Section 3. Finally, we conclude the paper in Section 4.

2. Description of Models

2.1. Baseline Predictor for Item-Based CF Normalization. A general neighborhood-based collaborative filter recommendation using BLP normalization is defined as follows:

$$\hat{r}_{ui} = b_{ui} + \Upsilon_k(u,i). \quad (1)$$

$\Upsilon_k(u,i)$ is rating predictor based on the k nearest neighbors. b_{ui} is the baseline predictor, which is always defined as

$$b_{ui} = \mu + b_i + b_u. \quad (2)$$

Denote by μ the average ratings. The parameters b_i and b_u indicate the observed deviations of item i and user u, respectively, from the average.

For item-based CF, we do not use the user biases due to using the similar items as neighbors. So the BLP in item-based CFS is

$$b_{ui} = \mu + b_i. \quad (3)$$

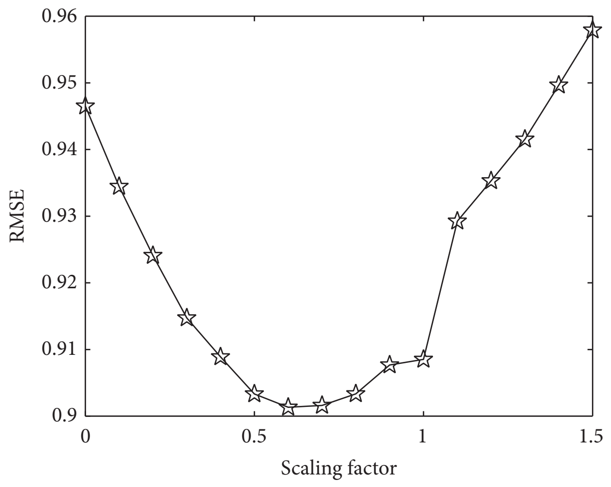

Figure 1: Impact of unified scaling factor on RMSE.

$\Upsilon_k(u,i)$ is replaced by the following formula:

$$\Upsilon_k(u,i)_{ib} = \frac{\sum_{j\in S(i,k)\cap N(u)} w_{ij}\left(r_{uj} - b_{uj}\right)}{\sum_{j\in S(i,k)\cap N(u)} \left|w_{ij}\right|}. \quad (4)$$

$S(i,k)$ is the set of the most similar k items to the item u, and $N(u)$ is the set of items the user u has rated.

There are many different similar weight functions. In this paper, we use two popular ones, Cosine and Pearson's Correlation, the details of which are defined, respectively, as

$$\begin{aligned}\operatorname{Cos}\left(w_{i,j}\right) &= \frac{\sum_{u\in U} r_{ui}\cdot r_{uj}}{\sqrt{\sum_{u\in U} r_{ui}^2 \cdot \sum_{u\in U} r_{uj}^2}},\\ \operatorname{PC}\left(w_{i,j}\right) &= \frac{\sum_{u\in U}\left(r_{ui} - b_{ui}\right)\cdot\left(r_{uj} - b_{uj}\right)}{\sqrt{\sum_{u\in U}\left(r_{ui} - b_{ui}\right)^2 \cdot \sum_{u\in U}\left(r_{uj} - b_{uj}\right)^2}}.\end{aligned} \quad (5)$$

2.2. Motivation of Scaling Baseline Predictor. The baseline predictor can introduce some information which is independent of neighborhood influence, but it is always set as a constant. However, we found that slightly scaling the baseline predictor could get a better predicting accuracy. But using a single scaling factor for μ, b_i, and b_u is not a good idea. Figure 1 shows an example where we can decrease the RMSE when scaling b_{ui} (e.g., αb_{ui}) on a small MovieLens dataset.

From Figure 1, the best scaling factor is 0.6, at which we can get the lowest RMSE. However, from another perspective, such as Top$N$ measure, using the same scaling factor 0.6 is not a good choice. Figure 2 shows that scaling BP could not improve the precision and recall.

For the recommendation systems, Top$N$ measure is more important than RMSE. To improve both RMSE and Top$N$ measure, we should not use the same scaling factor for the parameters in b_{ui}:

$$\hat{b}_{ui} = \alpha\mu + \gamma b_i. \quad (6)$$

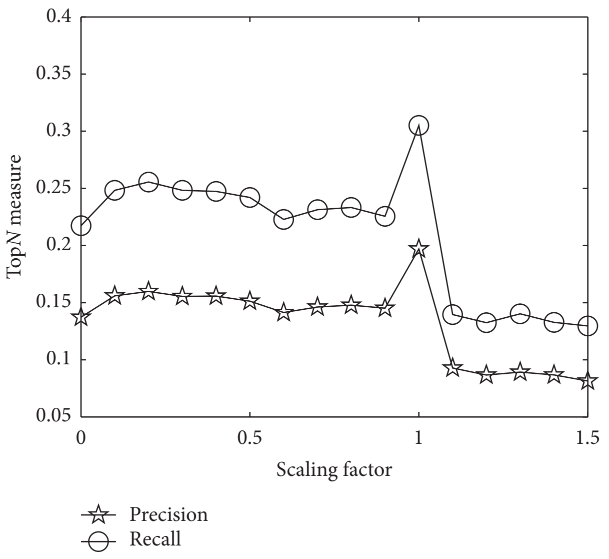

FIGURE 2: Impact of unified scaling factor on precision and recall.

Determining these three parameters is very difficult, but, unlike matrix factorization models, NBCFs can also not train the unknown parameters. In this paper, we provide several context-aware scaling factors. Before describing the details, we first change (6) to another representation. Actually, the baseline predictor can be also described as

$$b_{ui} = \mu + \frac{\sum_u (r_{ui} - \mu)}{|U_i|}. \tag{7}$$

U_i is the set of users rating item i. The scaling version of baseline predictor can be considered as

$$\hat{b}_{ui} = \alpha\mu + \frac{\sum_u (r_{ui} - \mu)}{(|U_i| + \beta)} + \frac{\sum_i (r_{ui} - \mu)}{(|I_u| + \beta)}. \tag{8}$$

Here, we use the denominator β to control the scaling factors, and hence $\gamma = |U_i|/(|U_i|+\beta)$. In fact, β is the Bayesian mean damping term [24]. It biases means toward the global mean $\alpha\mu$. Our task is to determine α and β according to the context information.

The recommendation system is a very special machine learning research. The user-item matrix is always too sparse. When data is sparse, we need other sources of knowledge to help the machine learning algorithm. Mining the context information is a way of adding knowledge to the recommendation system algorithms.

2.3. Context-Aware Scaled Baseline Predictors. We consider several context situations to determine the scaling baseline predictors: ratings distribution, categories distribution, timestamp distribution, and links distribution. At first, we denote by I the set of all the items and by U the set of all the users.

The rating distribution aware (RDA) method scales the baseline predictors in terms of ratings distribution. The values of ratings are usually discrete. Denote by $V = \{v_1, v_2, \ldots, v_m\}$ the set of possible rating values, where $v_1 < v_2 < \cdots < v_m$.

Denote by $N(v_i)$ the set of rating records of which the value is v_i:

$$N(v_i) = \{\langle u, i, r\rangle\},$$
$$N(v_i) \cdot r = v_i. \tag{9}$$

u is the user, i represents the item, and r means the rating of i rated by u. Also, $U(v_i)$ denotes the set of users whose ratings contain v_i, and $I(v_i)$ denotes the set of items which are rated using the value v_i. Now we sort all $N(v_i)s$, and let V_{dsc} be the set of $N(v_i)s$ order by descent according to the size of sets: $V_{\text{dsc}} = \{N(v_j)_1, N(v_h)_2, \ldots, N(v_q)_m\}$, where $|N(v_j)_1| \geq |N(v_h)_2| \geq \cdots \geq |N(v_q)_m|$, $v(i) \in V$. Denote by E all the rating records. The scaling factors of RDA are evaluated as

$$\alpha = \frac{\sum |N(v_i)_d|}{|E|}, \quad d \leq k,$$
$$\beta = \frac{\sum |N(v_i)_d|}{|\bigcup U(N(v_i)_d \cdot r)|}, \quad d \leq k. \tag{10}$$

Here, we use the k largest $N(v_i)s$. If the sizes of some sets are equal and the number of candidates is larger than k, we randomly select the sets of the same size.

Like RDA, the category distribution aware (CDA) method scales the baseline predictors in terms of category distribution. The items in recommendation system always have some labels, indicating some special attributes. In the MovieLens, the movies have some labels of genres. Each label corresponds to a category, and each item may belong to at least one category.

Suppose we have s categories, and denote by C the set of these s different categories, where $C = \{c_1, c_2, \ldots, c_s\}$. Denote by $I(c_i)$ the set of items belonging to c_i. I_{dsc} is a descent ordered set according to the size of set: $I_{\text{dsc}} = \{I(c_j)_1, I(c_h)_2, \ldots, I(c_q)_s\}$, where $|I(c_j)_1| \geq |I(c_h)_2| \geq \cdots \geq |I(c_q)_s|$, $c_i \in C$. For CDA, the scaling factors are expressed as

$$\alpha = \frac{|\bigcup I(c_i)_d|}{|I|}, \quad d \leq k,$$
$$\beta = \frac{\sum |I(c_i)_d|}{|\bigcup I(c_i)_d|} \cdot |C|, \quad d \leq k. \tag{11}$$

Note that to determine β we use $\sum |I(c_i)_d|$ as the numerator and $|\bigcup I(c_i)_d|$ as the denominator. The difference is that the items always belong to multiple categories.

There is always a timestamp record for each rating. The timestamp distribution aware (TDA) method scales baseline predictor in terms of timestamp distribution. Suppose that the element of E is a 4-tuple, where $e_l = \langle u, i, r, t\rangle \in E$. The meanings of u, i, and r are the same as in $N(v_i)$. t is just the timestamp when u rated i by the score r. The format of t is usually a Unix timestamp. We change t to "yy-mm-dd" format $\hat{t}$. That means the base unit of time is the day, and now $e_l = \langle u, i, r, \hat{t}\rangle$.

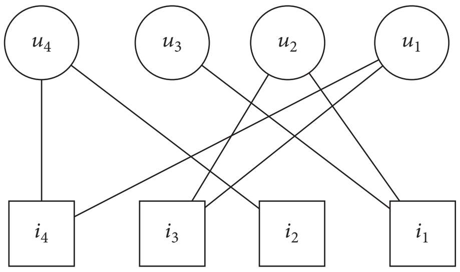

Figure 3: Example of rating network.

Let $T(\hat{t}_i)$ be the set of rating records, of which the reduced timestamp is $\hat{t}$. Like the previous two methods, we create a descent ordered set $T_{\text{dsc}} = \{T(\hat{t}_j)_1, T(\hat{t}_h)_2, \ldots, T(\hat{t}_q)_z\}$ according to the size of $T(\hat{t}_i)$, where $|T(\hat{t}_j)_1| \geq |T(\hat{t}_h)_2| \geq \cdots \geq T(\hat{t}_q)_z$.

We select the first g elements of T_{dsc} to compose another truncated set $T^g_{\text{dsc}} = \{T(\hat{t}_j)_1, T(\hat{t}_h)_2, \ldots, T(\hat{t}_y)_g\}$, $g \leq z$, $T^g_{\text{dsc}} \subseteq T_{\text{dsc}}$. Denote by $U((\hat{t}_i)_d)$ the set of distinct users of the rating records belonging to $T((\hat{t}_i)_d)$. The scaling factors of TDA are expressed as

$$\alpha = \frac{(1/g)\sum_{d\leq g} |T(\hat{t}_i)_d|}{(1/k)\sum_{d\leq k} |T(\hat{t}_i)_d|}, \quad k \leq g,$$
$$\beta = \sum_{d\leq k} |U((\hat{t}_i)_d)|, \quad k \leq g. \tag{12}$$

The links distribution aware (LDA) method scales baseline predictor in terms of links distribution. The links mean the relationship between users and items, which make up a rating network. Any pairs of users have no link, and any pairs of items also have no link. Equation (13) and Figure 3 show an example of rating network:

$$\begin{array}{c|cccc} & i_1 & i_2 & i_3 & i_4 \\ \hline u_1 & 0 & 0 & 1 & 1 \\ u_2 & 1 & 0 & 1 & 0 \\ u_3 & 1 & 0 & 0 & 0 \\ u_4 & 0 & 1 & 0 & 1 \end{array} . \tag{13}$$

Only when the rating between u and i is larger than or equal to r_T can we connect u and i. The degree of the user u is expressed as $\phi^U(u)$ and $\phi^I(i)$ for the item i. We create two descent ordered sets U_{dsc} and I_{dsc} according to the degrees. It is obvious that $U_{\text{dsc}} = U$ and $I_{\text{dsc}} = I$. But, for convenience, we use different symbols. That is, $U_{\text{dsc}} = \{\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_m\}$ and $I_{\text{dsc}} = \{\hat{i}_1, \hat{i}_2, \ldots, \hat{i}_n\}$. There is a unique mapping from $u_i \in U$ to $\hat{u}_i \in U_{\text{dsc}}$ and from $i_i \in I$ to $\hat{i}_j \in I_{\text{dsc}}$. For U_{dsc} and I_{dsc}, we have $\phi^U(\hat{u}_1) \geq \phi^U(\hat{u}_2) \geq \cdots \geq \phi^U(\hat{u}_m)$ and $\phi^I(\hat{i}_1) \geq \phi^I(\hat{u}_i) \geq \cdots \geq \phi^I(\hat{i}_n)$. We put the ordered degrees of users and items into two sets, respectively: $D^U = \{D_u(1), D_u(2), \ldots, D_u(m)\}$ and $D^I = \{D_i(1), D_i(2), \ldots, D_i(n)\}$, where $D_u(1) \geq D_u(2) \geq \cdots \geq D_u(m)$ and $D_i(1) \geq D_i(2) \geq \cdots \geq D_i(n)$.

Table 1: The meanings of different methods' names.

Name	Baseline predictor	Similarity
NoBP-Cos/PC	No	Cosine/Pearson's Correlation
USBP-Cos/PC	Unscaled BP	Cosine/Pearson's Correlation
RDA-Cos/PC	RDA	Cosine/Pearson's Correlation
CDA-Cos/PC	CDA	Cosine/Pearson's Correlation
TDA-Cos/PC	TDA	Cosine/Pearson's Correlation
LDAU-Cos/PC	LDAU	Cosine/Pearson's Correlation
LDAI-Cos/PC	LDAI	Cosine/Pearson's Correlation

For LDA, we have two ways of evaluating the scaling factors. When considering the degrees of the users, the method is called LDAU, which is expressed as

$$\alpha = \frac{(1/g)\sum_{d\leq g} |D_u(d)|}{(1/k)\sum_{d\leq k} |D_u(d)|}, \quad k \leq g \leq m,$$
$$\beta = \frac{(1/g)\sum_{d\leq g} |D_u(d)|}{(1/m)\sum_{d\leq m} |D_u(d)|}, \quad g \leq m. \tag{14}$$

Also, when considering the degrees of the items, the method is called LDAI, and the scaling factors are expressed as

$$\alpha = \frac{(1/g)\sum_{d\leq g} |D_I(d)|}{(1/k)\sum_{d\leq k} |D_I(d)|}, \quad k \leq g \leq n,$$
$$\beta = \frac{(1/g)\sum_{d\leq g} |D_I(d)|}{(1/n)\sum_{d\leq n} |D_I(d)|}, \quad g \leq n. \tag{15}$$

Unlike the other methods, LDA controls α and β using different distributions. For α, we use the top k and top g degrees, but for β, we use the top g degrees and the average degree.

3. Experiments

3.1. Experimental Settings. We use a MovieLens latest dataset in our experiments, including 100,000 ratings and 6,100 tag applications applied to 10,000 movies by 700 users [25]. There are four files for each dataset: links, movies, ratings, and tags. We use these files to get different context information. We compare several different methods in our experiments, the names and meanings of which are shown in Table 1.

The total methods compared are defined in Table 1. There are two similarity weight functions in our experiments: Cosine and Pearson's Correlation. The neighborhood sizes of item-based models are all set to 20, while they are 100 for user-based models. Values of k in (11)~(15) are the same, 6 in default. The values of g are also the same for these different methods, 20 in default. We randomly split the dataset into 5 parts and use cross-validation to train and test the models.

For top$N$ metric (e.g., precision and recall), we randomly select 100 items on the testing as the candidates, excluding the ones appearing in the training. Only the items rated above 3.5 (including 3.5) are recommended.

Table 2: Values of scaling factors.

Method	α	β
RDA	0.8125	32.3833
CDA	0.8964	29.6461
TDA	0.7598	37
LDAU	0.6544	12.5643
LDAI	0.8083	29.8041

Table 3: Experimental results using Cosine.

Method	Precision	Recall	RMSE	MAE	nDCG
NoBP-Cos	12.99%	20.62%	0.9465	0.9410	0.7237
USBP-Cos	19.82%	30.78%	0.9085	0.6939	0.9389
RDA-Cos	25.34%	39.01%	0.8984	0.6889	0.9485
CDA-Cos	25.05%	39.57%	0.8973	0.6869	0.9490
TDA-Cos	25.60%	40.03%	0.8999	0.6901	0.9488
LDAU-Cos	26.15%	40.80%	0.9003	0.6904	0.9495
LDAI-Cos	25.30%	39.45%	0.8979	0.6886	0.9485

The neighborhood collaborative filter models always incur a high memory cost. So we use a 16 GB RAM to run different NHCF algorithms.

3.2. Experimental Metrics. Five metrics are used in our experiments: precision, recall, RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), and nDCG (Normalized Cumulative Discounted Gain).

For a test dataset τ, denote by TP the set of recommend items which the users are really interested in, denote by FP the set of recommend items which the users are not interested in, denote by FN the set of not recommend items which the users are interested in, and denote by TN the set of not recommend items which the users are not interested in. The metrics of precision and recall are defined, respectively, as follows:

$$\text{precision} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FP}|},$$
$$\text{recall} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|}. \tag{16}$$

The recommendation system generates predicted ratings $\hat{r}_{ui}$ for a test set τ of user-item pairs (u, i) for which the true ratings r_{ui} are known. The RMSE and MAE between the actual ratings are given by

$$\text{RMSE} = \sqrt{\frac{1}{\tau} \sum_{(u,i)\in\tau} \left(\hat{r}_{ui} - r_{ui}\right)^2},$$
$$\text{MAE} = \sqrt{\frac{1}{\tau} \sum_{(u,i)\in\tau} \left(\hat{r}_{ui} - r_{ui}\right)}. \tag{17}$$

The recommendation systems always present to the user a list of recommendations, imposing a certain natural browsing order. In many cases, we are not interested in predicting an explicit rating or selecting a set of recommended items, as in the previous sections; rather we are interested in ordering items according to the user's preferences. nDCG is a measure from information retrieval, where positrons are discounted algorithmically. Assuming that each user u has a "gain" g_{ui} from being recommended an item i, the average Discounted Cumulative Gain (DCG) for a list of J items is defined as

$$\text{DCG} = \frac{1}{N} \sum_{u=1}^{N} \sum_{j=1}^{J} \frac{g_{ui_j}}{\max\left(1, \log_b j\right)}, \tag{18}$$

where the logarithm base is a free parameter, typically between 2 and 10. A logarithm with base 2 is commonly used to ensure that all positions are discounted. nDCG is just the normalized version of DCG:

$$\text{nDCG} = \frac{\text{DCG}}{\text{DCG}^*}, \tag{19}$$

where DCG* is the ideal DCG, the value of which ranges from 0 to 1. The larger the value is, the better the performance is.

3.3. Experimental Results. We change a little the format of the MovieLens dataset and import this dataset to a MySQL database. The coefficients of BLP can be conveniently calculated using some advanced SQL sentences. All of the coefficients of CASBLP methods are shown in Table 2.

The experimental comparison results are shown in Table 3 (using Cosine similarity) and Table 4 (using Pearson's Correlation). It seems that using Cosine is better than using Pearson's Correlation in our experiments. Maybe this is because even if each user has different personal rating scale, the rating matrix is too sparse to become the major issue. When data is sparse, Cosine is always a good choice.

From Table 3, we can see that when not using normalization scheme (NoBP) all of the metrics are much worse

Table 4: Experimental results using Pearson's Correlation.

Method	Precision	Recall	RMSE	MAE	nDCG
NoBP-PC	12.93%	20.34%	1.2229	0.9271	0.9109
USBP-PC	20.01%	30.35%	0.9355	0.7169	0.9348
RDA-PC	25.24%	38.38%	0.9603	0.7440	0.9306
CDA-PC	24.65%	36.90%	0.9483	0.7340	0.9311
TDA-PC	25.19%	38.40%	0.9709	0.7522	0.9300
LDAU-PC	26.65%	40.93%	0.9856	0.7590	0.9327
LDAI-PC	25.19%	38.34%	0.9596	0.7434	0.9307

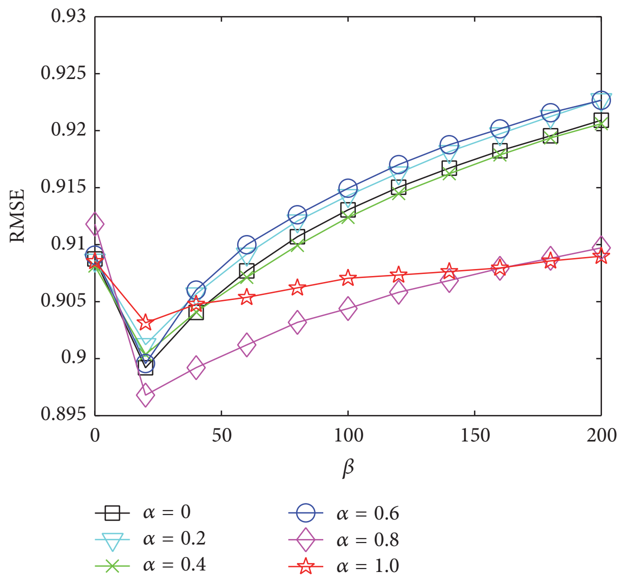

Figure 4: Impact of scaled factors on RMSE.

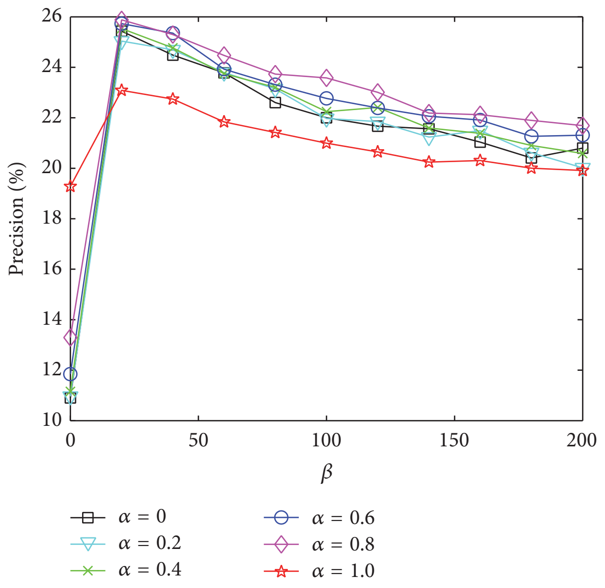

Figure 5: Impact of scaled factors on precision.

than the others. The unscaled BLP is even better than NoBP, in which the precision increases by about 7%, recall increases by more than 10%, and RMSE decreases by 4%. It is surprising that only using the simple unscaled BLP the MAE increases by 15% and the nDCG increases by more than 20%. Because recommendation order has a great commercial significance, the normalization is an important improvement in recommendation system. Our context-aware scaled BLP normalization schemes make further improvement, mainly on the precision and recall metrics. From both Tables 3 and 4, CASBLP normalization has almost the same RMSE, MAE, and nDCG as the USBP, sometimes even little worse than USBP. But, for a commercial recommendation system, what the users care about is whether the RSs recommend what they really need. The production selling would benefit from even a 1% improvement on precision or recall. The precision of our CASBLP schemes increases by about 5%, and the recall increases by about 8%, which is a great improvement from the commercial perspective.

An important problem is that the coefficients we used have optimal values. So we change α from 0 to 1 and β from 0 to 200 to see the changes of the performance. Figures 4–6

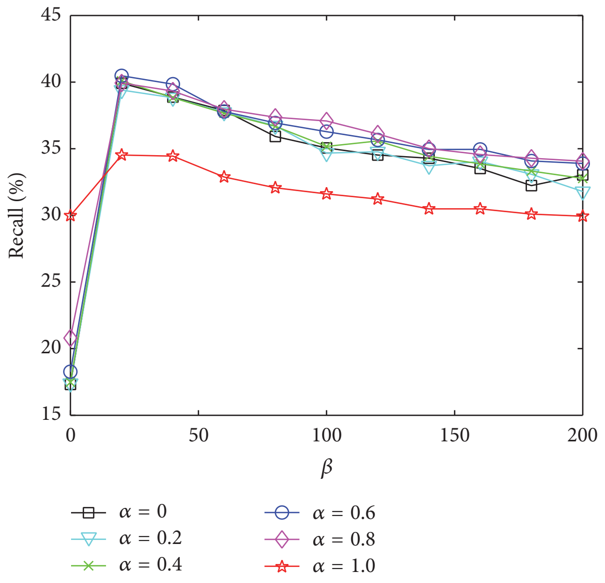

Figure 6: Impact of scaled factors on recall.

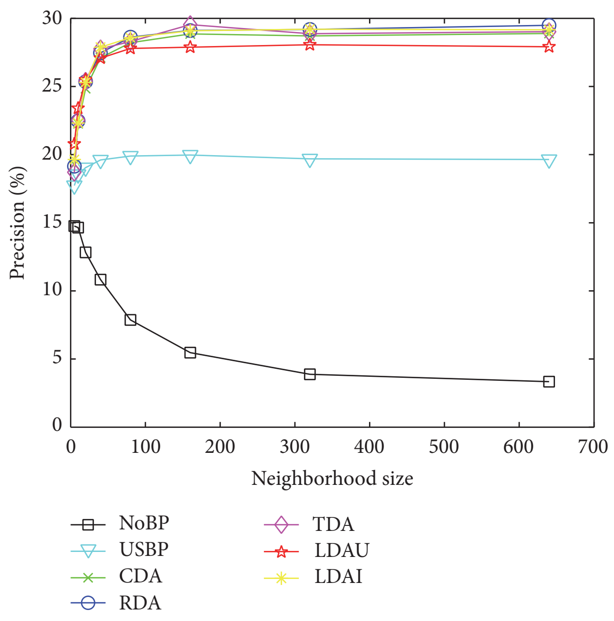

Figure 7: Impact of neighborhood size on precision.

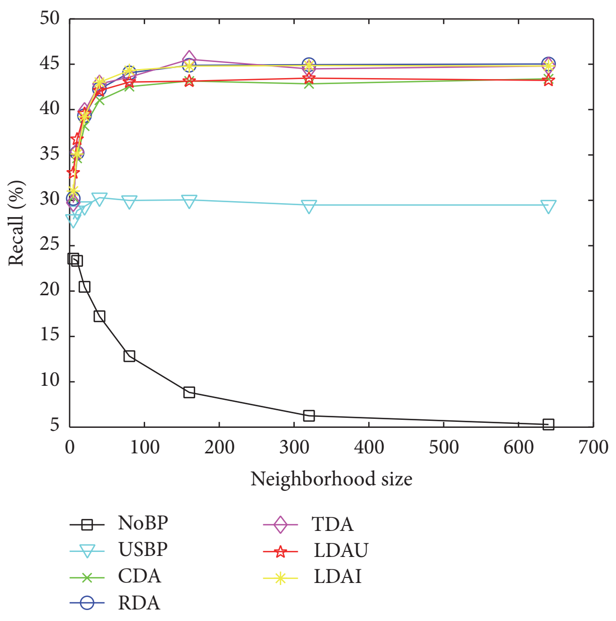

Figure 8: Impact of neighborhood size on recall.

show the impact of scaled factors on RMSE, precision, and recall, respectively.

For all these three metrics, the optimum of β is near 20, at which the RMSE is the lowest and the precision and recall are the highest. What is interesting is that any shrinking of α can improve precision and recall, even if we set α to zero. However, shrinking α would cause a slightly higher RMSE except at the value near 0.8.

This means that α can control the accuracy of the rating prediction, but when α has shrunk, β plays a crucial role in items recommendation. What causes this phenomenon is that maybe the mean rating is computed in terms of all the users, which involves the global information, while the biases are computed in terms of only very few similar neighbors, which involves the local information. For the personalized recommendation systems, the local information is much more important, and an ordinary average prediction has little meaning. That is why even if we set α to 0 and only using the item biases we can also get a passable prediction performance.

The neighbor size is an important factor in the neighborhood-based recommendation systems, for item-based or user-based ones. We increase the neighbor size geometrically from 5 to 320. Figures 7, 8, and 9 show the change of recommendation performance including precision, recall, and RMSE, respectively.

What we can see from Figure 9 is consistent with what we have concluded from Tables 3 and 4. Whether using scaled BLP or unscaled BLP, we can get similar RMSE, which are all much lower than the NoBP scheme. With the growth of the neighbor size, all the RMSE are trending toward stability.

What surprised us is the results of precision and recall. Both metrics are increasing until reaching the stable values with the growth of neighbor size except the NoBP scheme, the precision and recall of which decrease to the stable values. This is due to the fact that, maybe without normalization, the prediction lacks personalization and causes too many more decoys to choose from.

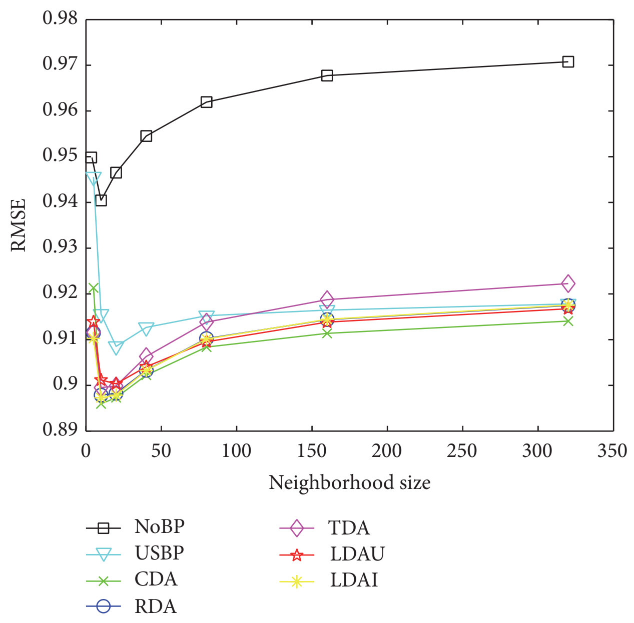

Figure 9: Impact of neighborhood size on RMSE.

Figures 7 and 8 also show the results which are consistent with Tables 3 and 4. Just slightly changing the coefficients of BLP, we can get higher precision and recall than unscaled BLP scheme and NoBP especially when using larger neighbor size.

4. Conclusions

Rating normalization is an important step when designing collaborative filter recommendation systems, especially for the item-based ones which play a key role in the domain of online commercial business. Using the baseline predictor for normalization considers both the global information and local information. Although we found that balancing them can improve the recommendation performance, there is no clear way of determining the weight of these two sources of information. In this paper, we proposed some context-aware scaled BLP schemes, which compute the weights of mean ratings and biases, respectively, in terms of different context information. What we concluded from the experiments not only verified the advantage of scaled BLP but also pointed out the different roles of each part of BLP. This paper only studied the BLP normalization of item-based collaborative filter system on a sole MovieLens dataset. The user-based and matrix factorization models actually are much different from item-based ones, the details of which we will explore in the future work using some different and larger recommendation dataset.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (no. 61602399 and no. 61502410) and Shandong Provincial Natural Science Foundation, China (no. ZR2016FB22).

References

[1] T.-Y. Ku, H.-S. Won, and H. Choi, "Service recommendation system for big data analysis," in *Proceedings of the 30th International Conference on Information Networking (ICOIN '16)*, pp. 317–320, January 2016.

[2] D. Zhang, T. He, Y. Liu, S. Lin, and J. A. Stankovic, "A carpooling recommendation system for taxicab services," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 254–266, 2014.

[3] L. Yao, Q. Z. Sheng, A. H. H. Ngu, J. Yu, and A. Segev, "Unified collaborative and content-based web service recommendation," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 453–466, 2015.

[4] Z.-S. Chen, J.-S. R. Jang, and C.-H. Lee, "A kernel framework for content-based artist recommendation system in music," *IEEE Transactions on Multimedia*, vol. 13, no. 6, pp. 1371–1380, 2011.

[5] W. Paireekreng, "Mobile content recommendation system for re-visiting user using content-based filtering and client-side user profile," in *Proceedings of the 12th International Conference on Machine Learning and Cybernetics (ICMLC '13)*, vol. 4, pp. 1655–1660, July 2013.

[6] X. Du, L. Huang, and Y. Du, "Improve the collaborative filtering recommender system performance by trust network construction," *Chinese Journal of Electronics*, vol. 25, no. 3, pp. 418–423, 2016.

[7] L. Wang, X. Meng, and Y. Zhang, "Applying HOSVD to alleviate the sparsity problem in Context-aware recommender systems," *Chinese Journal of Electronics*, vol. 22, no. 4, pp. 773–778, 2013.

[8] Y. Cai, H.-F. Leung, Q. Li, H. Min, J. Tang, and J. Li, "Typicality-based collaborative filtering recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 3, pp. 766–779, 2014.

[9] Y. Hu, Q. Peng, X. Hu, and R. Yang, "Time aware and data sparsity tolerant web service recommendation based on improved collaborative filtering," *IEEE Transactions on Services Computing*, vol. 8, no. 5, pp. 782–794, 2015.

[10] J. Wu, L. Chen, Y. Feng, Z. Zheng, M. C. Zhou, and Z. Wu, "Predicting quality of service for selection by neighborhood-based collaborative filtering," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 43, no. 2, pp. 428–439, 2013.

[11] X. Ma, C. Wang, Q. Yu, X. Li, and X. Zhou, "An FPGA-based accelerator for neighborhood-based collaborative filtering recommendation algorithms," in *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER '15)*, pp. 494–495, IEEE, September 2015.

[12] Y. El Madani El Alami, E. H. Nfaoui, and O. El Beqqali, "Toward an effective hybrid collaborative filtering: a new approach based on matrix factorization and heuristic-based neighborhood," in *Proceedings of the 1st International Conference on Intelligent Systems and Computer Vision (ISCV '15)*, pp. 1–8, March 2015.

[13] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[14] J. Zou and F. Fekri, "A belief propagation approach to privacy-preserving item-based collaborative filtering," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 7, pp. 1306–1318, 2015.

[15] N. Chang and T. Terano, "Improving the performance of user-based collaborative filtering by mining latent attributes of neighborhood," in *Proceedings of the International Conference on Mathematics and Computers in Sciences and in Industry (MCSI '14)*, pp. 272–276, September 2014.

[16] Z. Jia, Y. Yang, W. Gao, and X. Chen, "User-based collaborative filtering for tourist attraction recommendations," in *Proceedings of the IEEE International Conference on Computational Intelligence and Communication Technology (CICT '15)*, pp. 22–25, February 2015.

[17] M. R. N. Ranjbar, M. G. Tadesse, Y. Wang, and H. W. Ressom, "Bayesian normalization model for label-free quantitative analysis by LC-MS," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 12, no. 4, pp. 917–927, 2015.

[18] J. Liu and G. Deng, "A new-user cold-starting recommendation algorithm based on normalization of preference," in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '08)*, pp. 1–4, October 2008.

[19] R. Jin, L. Si, and C. Zhai, "Preference-based graphic models for collaborative filtering," in *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI '03)*, pp. 329–333, San Francisco, Calif, USA, 2003.

[20] R. Jin, L. Si, C. Zhai, and J. Callan, "Collaborative filtering with decoupled models for preferences and ratings," in *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM '03)*, pp. 309–316, November 2003.

[21] J. Li, P. Feng, and J. Lv, "ICAMF: improved context-aware matrix factorization for collaborative filtering," in *Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '13)*, pp. 63–70, November 2013.

[22] M. Tang, Z. Zheng, G. Kang, J. Liu, Y. Yang, and T. Zhang, "Collaborative web service quality prediction via exploiting matrix factorization and network map," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 126–137, 2016.

[23] X. Luo, M. Zhou, H. Leung et al., "An incremental-and-static-combined scheme for matrix-factorization-based collaborative filtering," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 333–343, 2016.

[24] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, "Collaborative filtering recommender systems," *Foundations and Trends in Human-Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2010.

[25] GroupLens, MovieLens Latest Datasets, January 2016, http://grouplens.org/datasets/movielens/.