Collaborative Filtering with Implicit Feedbacks by Discounting Positive Feedbacks

Kento Kawai
Graduate School of Systems and Information Engineering
University of Tsukuba
Tsukuba, Japan
Email: kentkawai@kde.cs.tsukuba.ac.jp

Hiroyuki Kitagawa
Faculty of Engineering, Information and Systems
University of Tsukuba
Tsukuba, Japan
Email: kitagawa@cs.tsukuba.ac.jp

***Abstract*—Recommender Systems are indispensable to provide personalized services on the Web. Recommending items which match a user's preference has been researched for a long time, and there exist a lot of useful approaches. Especially, Collaborative Filtering, which gives recommendation based on users' feedbacks to items, is considered useful. Feedbacks are categorized into explicit feedbacks and implicit feedbacks. In this paper, Collaborative Filtering with implicit feedbacks is addressed. Explicit feedbacks are feedbacks provided by users intentionally and represent users' preferences for items explicitly. For example, in Netflix, users can rate movies on a scale of 1-5, and, based on these ratings, users can receive movie recommendation. On the other hand, implicit feedbacks are collected by the system automatically. In Amazon.com, products that users buy and click are used for recommendation. While Collaborative Filtering with explicit feedbacks has been a central topic for a long time, implicit feedbacks have become a more and more important research topic recently because these are easier to obtain and more abundant than explicit feedbacks. However, implicit feedbacks are often noisy. They often contain feedbacks which do not represent users' real preferences for items. Our approach addresses to this noise problem. We propose three discounting methods for observed values in implicit feedbacks. The key idea is that there is hidden uncertainty for each observed feedback, and effects by observed feedbacks of much uncertainty are discounted. The three discounting methods do not need additional information besides ordinary user-item feedbacks pairs and timestamps. Experiments with huge real-world datasets confirm that all of the three methods contribute to improving the performance. Moreover, our discounting methods can easily be combined with existing methods and improve the recommendation accuracy of existing models.**

I. Introduction

Recommender Systems are indispensable when providing personalized services on the Web. For example, Amazon.com[1] recommends products based on individual preferences. Since this has been an area of importance for a long time, many useful approaches have been proposed.

These existing recommendation approaches can be divided into three categories: Content-based Methods [9], Collaborative Filtering Methods [22], and hybrids of the two [10]. Content-based Method is based on the users' profiles and/or items' contents. On the other hand, Collaborative Filtering Method is based solely on past users' feedbacks on items. Collaborating Filtering Method is considered more useful than Content-based Method because it does not require domain knowledge, which is the knowledge about users and items in the domain. Also, Collaborative Filtering Method can generate unexpected recommendation which Content-based Method cannot generate. Feedbacks are categorized into *explicit feedbacks* and *implicit feedbacks* [1]. Explicit feedbacks are the feedbacks which users give to items intentionally and represent preferences of users to items explicitly. Feedbacks contain both positive feedbacks and negative feedbacks. One of the explicit feedbacks often used in recommender systems is rating. For example, in Netflix[2], users can rate movies with multi-class values on a scale of 1-5 (1 is the most negative and 5 is the most positive), and, based on these ratings, users can receive movie recommendation. On the other hand, implicit feedbacks are feedbacks which are collected by the system automatically and represent preferences of users to items implicitly. These feedbacks contain only positive values of one class. For example, implicit feedbacks such as products that users buy, click, and bookmark can be one-class positive values. Many approaches for recommendations based on explicit feedbacks have been proposed, but fewer studies have been done on collaborative filtering with implicit feedbacks due to its inherent difficulty. However, implicit feedbacks such as purchase history and access log can be automatically collected by the system, and they are easier to collect, whereas explicit feedbacks are given by users to items intentionally. Thus, collaborative filtering with implicit feedbacks has become a more and more important research topic recently.

This paper addresses the recommendation problem with implicit feedbacks. This problem is considered to be more difficult than the recommendation problem with explicit feedbacks. Many existing works assume that there are only positive feedbacks to items in observed implicit feedbacks. For example, when we treat purchase history, the observed

[1]http://www.amazon.com

[2]http://www.netflix.com

978-1-5090-2179-6/16 $31.00 © 2016 IEEE
DOI 10.1109/BigMM.2016.30

values are signals of purchased items, and the missing values are signals of non-purchased items. They assume that the former values suggest that the user likes the items, but the latter values do not necessarily indicate that the user dislikes the items. To address this problem, many existing approaches try to find possible negative values hidden in missing values. But, even in observed feedbacks, there may exist feedbacks which do not represent users' positive preferences for items. For example, in the case of real click log, users may have clicked items since they looked attracting, but found that they actually did not match their preferences. We call this noise problem in observed feedbacks. Our approach addresses this noise problem by proposing three discounting methods for observed (positive) one-class values. We assume that there is hidden uncertainty for each observed (positive) feedback. We discount effects by observed (positive) feedbacks of much uncertainty. The three discounting methods do not need additional information besides ordinary user-item feedbacks pairs and timestamps. Our discounting methods for positive feedbacks can easily be combined with existing methods for treating missing values and improve the recommendation accuracy of existing models.

The rest of the paper is organized as follows. Section II introduces related works. Section II-B describes our approach to discounting observed (positive) values. Section III shows the experimental results comparing our proposal with the baseline model. Section IV concludes the paper and discusses future work.

II. Related Work

A. Collaborative Filtering with Explicit Feedbacks

First, we discuss existing Collaborative Filtering methods with explicit feedbacks. Collaborative Filtering with explicit feedbacks that both positive and negative feedbacks are observed in the dataset. The Collaborative Filtering methods can be divided into the memory-based method, the model-based method and the combination of the two [13]. The memory-based method includes the Neighborhood method [5], which calculates the similarity of the users or items. The model-based method includes the Matrix Factorization model [15], the Probabilistic model [6] and Cluster-based model [4]. The Matrix Factorization model [15] is considered the most useful approach, which achieved the highest recommendation accuracy in the Netflix Prize [2]. This approach is based on the idea that there are latent factors which represent the user-item preference relationships between users and items, and unknown preferences can be predicted using latent factors and the relationship between users-latent factors and items-latent factors. The biggest problem in Collaborative Filtering is the sparseness of observed values. It means feedbacks are observed in very small portion of all possible user-item pairs. However the Matrix Factorization model is known to work better than other models even if the data is sparse.

B. Collaborative Filtering with Implicit Feedbacks

Here, we discuss existing Collaborative Filtering methods with implicit feedbacks. Basically, a dataset with implicit feedbacks consists of user-item pairs where the user provided feedbacks to the item. Often timestamps are also provided. Existing works for Collaborative Filtering with implicit feedbacks assume that implicit feedbacks are observed as one-class positive feedbacks and missing values do not indicate the negative feedbacks. Thus, existing Collaborative Filtering methods with explicit feedbacks can not be directly applied to the dataset with implicit feedbacks because they require both positive and negative feedbacks in the dataset. To address this problem, many existing approaches try to find possible negative values hidden in missing values.

Pan et al. [7], [8] employed a weighted Matrix Factorization model. They initially filled all missing values with negative values, and assign weights to discount the relative contribution of each value to prediction. The weights are determined based on the number of items to which a user provided feedback, or the number of users who gave feedbacks for an item, or given uniformly. This approach has the problem of running time. Basically, the Matrix Factorization model can save computation for missing values, but the weighted Matrix Factorization model must cope with all elements even when the original dataset is given very sparse. Pan et al. [7], [8] also proposed a sampling-based method. This approach samples only a part of the missing values and replace them with negative values. Three kinds of sampling methods are proposed: User-oriented sampling, Item-oriented sampling and Uniform sampling. User-oriented sampling assumes that the number of negative values hidden in missing values and the amount of past feedbacks given by a user are related. Item-oriented sampling assumes that the number of negative values hidden in missing values and the amount of past feedbacks given for an item are related. In this work, this method is combined with our proposals. Also, Zheng et al. [16] proposed an extension of the weighted Matrix Factorization model. They incorporates similarity matrices over items and users to the weighted Matrix Factorization model. Sindhwani et al. [26] [25] proposed the joint model of the Matrix Factorization model and non-negative Matrix Factorization model which classify missing values into positive and negative feedbacks. Packet and Koenigstein [17] proposed a bayesian generative model which predicts the probability with which missing values are converted to negative feedbacks. There exist other approaches that use the auxiliary data to treat missing values [18] [19] [20] [28].

As mentioned here, many approaches exist to find possible negative values hidden in missing values. However, they all treat observed (positive) feedbacks equally. As mentioned in

Section I, all of them do not actually represent users' positive preferences. Existing work [12] addressed this problem by estimating the likeliness of each feedback being a noise, which is called confidence. They used information of how many times a user gives feedbacks to an item. This approach assumes availability of auxiliary information of how many times a user gives feedbacks to an item, besides the implicit feedback dataset.

In this paper, we propose a method to address this problem without using auxiliary information. Our approach can be combined with existing models which find possible negative values in missing values.

C. Preliminary

Our method is an extension of the Matrix Factorization Model [15]. First, we discuss details of this model. The goal of the Matrix Factorization Model is to predict a user's preference for items via observed feedbacks. The Matrix Factorization Model assumes that latent factors exist between users and items, and unknown preferences can be predicted using latent factors and the relationship between users-latent factors and items-latent factors. Here, we use $U \times I$ matrix R to denote the observed user-item feedbacks, where U is the number of users, I is the number of items, and each entry R_{ui} represents the feedback from user u to item i if a feedback is observed, and R_{ui} is absent otherwise. Matrix Factorization Model tries to decompose R into $U \times K$ matrix P and $I \times K$ matrix Q to approximate R, where K is the number of latent factors.

$$R \approx PQ^T \tag{1}$$

By the decomposition, we get K-dimension vector P_u which represents user u's relevance to each latent factor and K-dimension vector Q_i which represents item i's relevance to each latent factor. This approximation is attained by solving the minimization problem with the regularizer to avoid overfitting as follows:

$$min \sum_{(u,i) \in Ob} (R_{ui} - P_u \cdot Q_i^T)^2 + \lambda(\|P\|^2 + \|Q\|^2). \tag{2}$$

Here, Ob is the set of user-item pairs where feedbacks are observed and $\lambda(\|P\|^2 + \|Q\|^2)$ is the regularizer, where λ is the parameter which controls the degree of avoid overfitting. This minimization problem is a non-convex problem, but there are two types of optimization methods proposed to address this problem: Stochastic Gradient Descent [15] and Alternating Least Squares [24]. Especially, we use Stochastic Gradient Descent to optimize in this work. With this optimization method, the optimization is achieved by updating P and Q for each user-item pair $(u, i) \in Ob$ as follows:

$$P_u \longleftarrow P_u + \gamma \times (e_{ui} Q_i - \lambda P_u) \tag{3}$$

$$Q_i \longleftarrow Q_i + \gamma \times (e_{ui} P_u - \lambda Q_i) \tag{4}$$

Here, $e_{ui} = P_u \cdot Q_i^T - R_{ui}$ gives the error between $P_u \cdot Q_i^T$ and R_{ui}. γ is the parameter which controls the learning rate of this optimization. This update process repeats until the objective function in (2) is converged. After the optimization, with the matrixes P and Q, the absent elements in R is estimated by calculating PQ^T. Here, we denote the complete $U \times I$ matrix as $\hat{R}$ where $\hat{R}_{ui} = P_u Q_i^T$. With the complete vector $\hat{R}_u$, we can determine which items to recommend to user u.

In this work, a dataset which contains one-class values is given. In this case, $R_{ui} = 1$ if the feedback from user u to item i is observed. If we apply existing works which treat missing values as mentioned in Section II-B, we can fill some or all of missing (unobserved) feedbacks with $R_{ui} = 0$.

D. Matrix Factorization Model with Discounting

We propose three discounting methods for observed values in implicit feedbacks. Here, we use common formulation for the three methods. We use the $U \times I$ discounting matrix W to represent the discounting ratio for each element of observed feedbacks in matrix R. Then, the discounted observed feedback from user u to item i is $W_{ui} R_{ui}$, where $0 < W_{ui} \leq 1$. After the discounting process, the original $U \times I$ matrix R is converted to $R_D = W \odot R$, where $\odot$ stands for element-wise multiplication. Then the Matrix Factorization Model is applied to R_D rather than R. In the next subsection, we explain how to determine the discounting matrix W.

E. Discounting Methods

We propose three different methods to determine the discounting matrix W: User-oriented, Item-oriented and Time-oriented Discounting. In this subsection, we use the term 'confidence' to represent how likely the given feedback represents the user's positive preferences for items rather than noises.

1) User-oriented Discounting: This discounting method is based on the assumption that the confidences of observed feedbacks depend on how many feedbacks a user provided for items. Intuitively, observed feedbacks by a user who gives feedbacks to a few items is more confident than those by a user who gives feedbacks to many items. With this intuition, W_{ui} is simply calculated as the inverse of the number of items a user u gives feedbacks to: $Ob^i(u)$. We use the non-zero parameter α to control the range of values in W. Then, this discounting method decides the discounting matrix W as follows:

$$W_{ui} = \frac{1}{Ob^i(u)^\alpha}. \tag{5}$$

Here, α is determined for each dataset because the range of $Ob^i(u)$ differs depending on datasets. For example, when $Ob^i(u)$ is very large for user u, W_{ui} gets very small. This means all the observed feedbacks by user u are discounted

to close to zero, which means negative feedbacks. By setting α to a small value, we can avoid this inconsistency. In our experiments, $\alpha < 0.2$ tends to get better performance.

2) Item-oriented Discounting: This discounting method is based on the assumption that the confidences of observed feedbacks depend on how many feedbacks an item gets from users. We assume that users tend to provide feedbacks to popular items do not necessarily represent positive preferences for items. This assumption is very similar to [23] which tries to recommend non-popular items. This method decides W_{ui} as the inverse of $Ob^u(i)$ which stands for the number of users who give feedbacks to an item i, with the non-zero parameter α.

$$W_{ui} = \frac{1}{Ob^u(i)^\alpha} \tag{6}$$

3) Time-oriented Discounting: This discounting method is based on the assumption that confidences of feedbacks decay as time passes. Intuitively, feedbacks given a long time ago may not represent the user's preferences for items when he/she receive recommendations. Here, we use the timestamp T_{ui} when user u gave the feedback to item i and $max(T)$, which is the maximum of T_{ui} in the given dataset. With the elapsed time $max(T) - T_{ui}$ between $max(T)$ and T_{ui} and the parameter α, this method decides W as follows:

$$W_{ui} = \frac{1}{(max(T) - T_{ui} + 1)^\alpha}. \tag{7}$$

The constant value 1 is added to avoid the denominator from being zero. In our experiments, we use the daily or the monthly time for T_{ui} by considering the time range of dataset. With the daily time T_{ui}, we give the same discounting for feedbacks which are provided at the same day. With the monthly time T_{ui}, we give the same discounting for feedbacks which are provided in the same month.

III. Experiments

In this section, we compare the recommendation performance of the baseline model and our proposed models through some types of experiments.

A. Models

In our experiments, we compare the following five models with three datasets.

Model Naive
: Uniform sampling method

Model User
: Naive + User-oriented Discounting

Model Item
: Naive + Item-oriented Discounting

Model Time(day)
: Naive + Time-oriented Discounting (daily)

Model Time(month)
: Naive + Time-oriented Discounting (monthly)

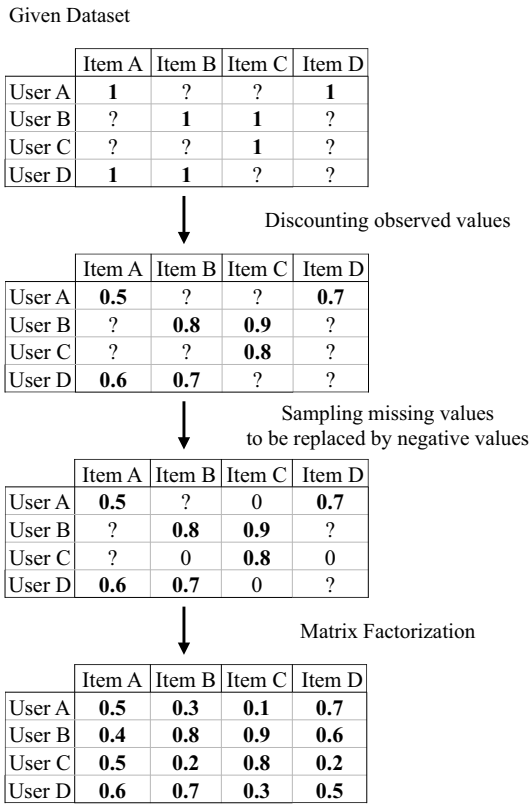

Figure 1. Combination of the Model Naive and Our Proposal

Here, the model **Naive** is the baseline model proposed in [7], which was explained in SectionII-B. The models **User**, **Item**, **Time(day)** and **Time(month)** are combinations of our proposed methods and the model **Naive**. The overall process of the models **User**, **Item**, **Time(day)** and **Time(month)** is outlined in Figure 1.

B. Datasets

In our experiments, we use three implicit feedback datasets. They are all real-world datasets (see Table I). The sparsity is the ratio of the number of observed feedbacks over the number of all possible user-item pairs. The number of observed feedbacks in datasets ranges from 188,483 to 4,413,834, and the number of all possible user-item pairs is between 10^9 and 10^{12}. We describe the features of each dataset below.

1) Rakuten Recipe: In the web service Rakuten Recipe[3], there are a lot of recipes written by users. Also, after a user used a recipe, he/she can write comments for the recipe. The dataset used here contains feedbacks which represent which

[3]http://recipe.rakuten.co.jp

Table I
DATASETS

Dataset	# of users	# of items	# of feedbacks	Sparsity
Rakuten Recipe	86,866	7,083	188,483	0.03%
Lastfm Song	933	1,084,621	4,413,834	0.4%
YOOCHOOSE	509,697	19,950	1,049,817	0.01%

user used which recipe. The recommendation task here is to recommend recipes which users will likely use.

2) Lastfm Song: Lastfm[4] provides the service of tracking each user's aggregated listening history of music over online streaming services and offline devices. The dataset we use here is collected by the author of [3] with Lastfm API. This data contains feedbacks which represent which user listen to which song. The recommendation task is to recommend songs which users would like to listen to. This dataset contains less users than other datasets, but contains much more items and feedbacks. Intuitively, recommendations tend to be more difficult as the number of items increases. Thus, the recommendation task using this dataset will be more difficult one.

3) YOOCHOOSE: This dataset is one of the e-commerce business dataset provided in RecSys Challenge 2015[5]. YOOCHOOSE[6] is a recommender system provider in e-commerce. This dataset represents which items were purchased during a session. We treat each session as a unique user. This dataset contains much more users and sparser than other datasets.

C. Evaluation Scheme

We use Mean Average Precision (MAP) as the recommendation evaluation scheme. MAP is commonly used in information retrieval to evaluate the relevance of the rankings generated for queries and many existing works applied this scheme to evaluate recommendation accuracy. In the recommendation process, for each user u, a recommendation item ranking is generated based on the decreasing order of $\hat{R}_u$. Then, the average precision (AP) for user u is calculated as

$$AP_u = \frac{\sum_{n=1}^{I} prec(n) \times pref(n)}{\#\ of\ preferred\ items} \tag{8}$$

where n is the position in the recommended item list, I is the number of items, $prec(n)$ is the precision at n, and $pref(n) = 1$ if the user gives a feedback to the n-th item in the test data, otherwise $pref(n) = 0$. This scheme measures the performance of how higher the model ranks items which user u give positive feedbacks in the test data. After calculating AP_u, MAP is obtained by taking the mean of AP_u over all users.

[4]http://last.fm
[5]http://recsys.yoochoose.net
[6]http://www.yoochoose.com

D. Experimental Setup

In all models, we randomly sample N missing values and replace them with negative values using the method proposed in [7]. The number of missing values N is fixed to be the same number of observed feedbacks. Because the experimental results depend on which missing values sampled, all experiments are repeated multiple times and each result is evaluated using a t-test with a 95% confidence interval. Experimental results are shown with the average and the standard deviation of the MAP score.

The number of latent factors is fixed to 10, which is almost the same value in existing works [25] [26] [21]. In our experiments, changing the number of latent factors did not significantly affect experimental results.

In this experiment, we divide each dataset into training data and test data. First, we ordered feedbacks by their timestamp, and divided them into 90% old and 10% new feedbacks. The former is used for training data and the latter is for test data.

E. Experimental Results

Figure 2 and Table III shows the experimental results. The results show that if the model **Naive** is combined with our methods, the recommendation accuracy tends to get higher. Especially, the model **Time(month)** achieves the highest improvement among proposals and the model **TIme(day)** follows. It suggests that confidences of feedbacks decay monthly rather than daily. The models **User** and **Item** also exceed the model **Naive**. It means that users' feedbacks for items are more certain when users give feedbacks to less popular items or users give less feedbacks to items. These two methods have advantages that they need only user-item pairs and do not need timestamps. With Lastfm Song and YOOCHOOSE datasets, MAP score tend to be smaller than with Rakuten Recipe dataset, because, when the number of items in dataset is higher, recommendation problem gets more difficult.

F. Additional Comparison

We also conducted more detailed experiments using the Rakuten Recipe dataset. We first ordered users by the number of feedbacks provided by the user in the training data, which is $Ob^i(u)$. Then all users are divided into

ten bins. Afterwards, the average MAP score for each bin is calculated. The results are shown in Table IV, where the number of provided feedbacks decreases from bin(1) to bin(10). Intuitively, the performance for each bin will decrease from bin(1) to bin(10) because the number of feedbacks decreases. However, Table IV does not coincide with this intuition because users who gave more feedbacks in training data tend to give more feedbacks in test data, and this tends to make the evaluation measure AP_u lower. In this result, the best discounting method for each bin is different. For bins (1) to (6), the model **Time(month)** performs the best, and the model **Item** performs the best for bin (7), (9), (10). This suggests that the former method is effective for users who give many feedbacks to items, and the latter method is effective for users who gave less feedbacks to items. Moreover, the latter method may be effective to the user cold-start problem [27].

We also divided users into ten bins by the average of item popularity that they provided feedbacks for, which is derived as the average of $Ob^u(i)$ over all items user u provides feedbacks for. The results are shown in Table V. The performance decreases from bin(1) to bin(10). This is related to the long-tail recommendation problem which was discussed in [23]. This result shows that, from bin(1) to bin(3), the model **Time(day)** performs the best, and, from bin(4) to bin(9), the model **Time(month)** performs the best. The model **Time(day)** works well to users who give feedbacks to popular items, and the model **Time(month)** works well to users who give feedbacks to non-popular items. Also, for the users who give feedbacks to extremely non-popular items, who are categorized to bin(10), the model **Item** improves the best. This reason may be that this method discounts the feedbacks which are provided for the popular items more and tends recommend non-popular items.

G. Additional Experiments

Although our experiments deployed the uniform sampling method [7], the proposed methods can be combined with other methods to treat missing values. To verify this, we conduct additional experiments using the Rakuten Recipe. The User-oriented sampling and Item-oriented sampling methods [7], which were explained in Section II-B, are combined with our methods. Table II show the results. In the experiments, our proposed methods improve the naive model for all sampling methods and Item-oriented sampling method exceeds the uniform sampling method. The combination of the Time-oriented discounting method and the Item-oriented sampling method yields the highest performance. The experimental results verify that the proposed discounting methods can be combined with different sampling methods and contribute to performance improvement.

IV. Conclusion and Future Work

We here proposed three different methods to discount the observed feedbacks: User-oriented Discounting, Item-oriented Discounting, and Time-oriented Discounting. Experiments using real-world large datasets show that all of our proposed methods improve the performance of the baseline model. Our methods are easily combined with existing models and can improve the recommendation performance. In particular, the Time-oriented Discounting method significantly improves the baseline model, indicating that feedbacks from users will turn to noises over time. Additionally, the User-oriented and Item-oriented Discounting methods can improve the performance although these methods use only information about which users gave feedbacks to which items. In other words, these methods can be used whenever user-item feedback data is available.

One of important future research issue is automatic selection of the best discounting method for each user.

Acknowledgment

This research was partly supported by the program "Research and Development on Real World Big Data Integration and Analysis" of MEXT and the Grant-in-Aid for Scientific Research (B) (#26280037) from JSPS. We used Rakuten Data in our experiments, which is provided by Rakuten, Inc, and distributed by National Institute of Informatics.

References

[1] G. Jawaheer, M. Szomszor, and P. Kostkova, "Comparison of implicit and explicit feedback from an online music recommendation service", Proc. 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems, pp. 47-51, 2010.

[2] R. Bell and Y. Koren, "Lessons from the Netflix Prize Challenge", SIGKDD Explorations 9 (2007), pp. 75-79.

[3] O. Celma, "Music Recommendation and Discovery in the Long Tail", Springer, 2010.

[4] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering", Proc. 5th IEEE International Conference on Data Mining, pp. 625-628, 2005.

[5] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. "GroupLens: An open architecture for collaborative filtering of Netnews", Proc. of the Conference on Computer Supported Cooperative Work, pp. 175-186, 1994.

[6] A. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: Scalable online collaborative filtering", Proc. of the 16th International Conference on World Wide Web, pp. 271-280, 2007.

[7] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering", Proc. 8th IEEE International Conference on Data Mining, pp. 502-511, 2008.

Table II
EXPERIMENTAL RESULTS OF COMBINING OTHER METHODS

Sampling Method		Naive	User	Item	Time(day)	Time(month)
Uniform	MAP	0.07444(±0.00176)	0.07855(±0.00160)	0.08191(±0.00130)	0.08391(±0.00204)	0.08445 (±0.00232)
	Gain	-	+5.52%	+10.03%	+12.73%	+13.45%
User-oriented	MAP	0.07191(±0.00162)	0.07791(±0.00251)	0.08429(±0.00116)	0.08219(±0.00097)	0.08227(±0.00173)
	Gain	-	+8.34%	+17.22%	+14.30%	+14.41%
Item-oriented	MAP	0.07817(±0.00188)	0.08089(±0.00252)	0.08407(±0.00193)	0.08510(±0.00171)	0.08710(±0.00213)
	Gain	-	+3.48%	+7.54%	+8.87%	+11.42%

[8] R. Pan and M. Scholz, "Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering", Proc.15th ACM SIGKDD International Conference on Knowledge discovery and data mining, pp. 667-676, 2009.

[9] M. Balabanovic and Y. Shoham, "Fab: Content-based, collaborative recommendation", Communications of the ACM, pp. 40(3):66-72, 1997.

[10] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin, "Combining content-based and collaborative filters in an online newspaper", In Proc. ACM SIGIR Workshop on Recommender Systems : Algorithms and Evaluation, 1999.

[11] A. Popescul, L. Ungar, D. Pennock, and S. Lawrence, "Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments", Proc. 17th Conference on Uncertainty in Artificial Intelligence, pp. 437-444, 2001.

[12] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets", Proc. 8th IEEE International Conference on Data Mining, pp. 263-272, 2008.

[13] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model", Proc. 14th ACM SIGKDD International Conference on Knowledge discovery and data mining, pp. 426-434, 2008.

[14] T. Joachims, "Optimizing search engines using clickthrough data", Proc. 8th ACM SIGKDD International Conference on Knowledge discovery and data mining, pp. 133-142, 2002.

[15] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems", In Computer, pp. 42(8):30-37, 2009.

[16] X. Zheng , H. Ding , H. Mamitsuka and S. Zhu, "Collaborative matrix factorization with multiple similarities for predicting drug-target interactions", Proc. 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1025-1033, 2013.

[17] U. Paquet and N. Koenigstein, "One-class collaborative filtering with random graphs", Proc. 22nd international conference on World Wide Web, pp. 999-1008, 2013.

[18] Y. Fang and L. Si, "Matrix co-factorization for recommendation with rich side information and implicit feedback", Proc. 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems, pp.65-69, 2011.

[19] N. Pappas and A. Popescu-Belis, "Sentiment analysis of user comments for one-class collaborative filtering over ted talks", Proc. 36th international ACM SIGIR conference on Research and development in information retrieval, pp. 773-776, 2013.

[20] Y. Li , J. Hu , C.X. Zhai and Y. Chen, "Improving one-class collaborative filtering by incorporating rich user information", Proc. 19th ACM international conference on Information and knowledge management, pp. 959-968, 2010.

[21] Y. Yao, H. Tong, G. Yan, F. Xu, X. Zhang, B. K. Szymanski, and J. Lu, "Dual-Regularized One-Class Collaborative Filtering", Proc. 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 759-768, 2014.

[22] D. Goldberg, D. Nichols, B. M. Oki and D. Terry, "Using Collaborative Filtering to Weave an Information Tapestry", In Communications of the ACM 35, pp. 61-70, 1992.

[23] H. Steck, "Item popularity and recommendation accuracy", Proc. 5th ACM conference on Recommender systems, pp. 125-132, 2011.

[24] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the Netflix Prize", Proc. 4th International Conference on Algorithmic Aspects in Information and Management pp. 337-348, 2008

[25] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic, "A family of non-negative matrix factorizations for one-class collaborative filtering", In ACM RecSys, 2009.

[26] V. Sindhwani, S. S. Bucak, J. Hu , and A. Mojsilovic, "One-Class Matrix Completion with Low-Density Factorizations", Proc. 10th IEEE International Conference on Data Mining, pp.1055-1060, 2010.

[27] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations", Proc. 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 253-260, 2002.

[28] X. Ning and G. Karypis, "Sparse linear methods with side information for top-n recommendations", Proc. 6th ACM conference on Recommender systems, pp. 155-162, 2012.

[29] X. Ning and G. Karypis, "SLIM: Sparse Linear Methods for Top-N Recommender Systems", Proc. 11th IEEE International Conference on Data Mining, pp.497-506, 2011.

Table III
EXPERIMENTAL RESULTS

Dataset		Naive	User	Item	Time(day)	Time(month)
Rakuten Recipe	MAP	0.07444(±0.00176)	0.07855(±0.00160)	0.08191(±0.00130)	0.08391(±0.00204)	0.08445 (±0.00232)
	Gain	-	+5.52%	+10.03%	+12.73%	+13.45%
Lastfm Song	MAP	0.01279(±0.00001)	0.01283(±0.00001)	0.01282(±0.00002)	0.01297(±0.00003)	0.01309(±0.00003)
	Gain	-	+0.27%	+0.21%	+1.40%	+2.32%
YOOCHOOSE	MAP	0.01591(±0.00198)	0.02650(±0.00298)	0.01906(±0.00124)	0.03074(±0.00191)	0.03848(±0.00320)
	Gain	-	+66.54%	+19.76%	+93.17%	+141.86%

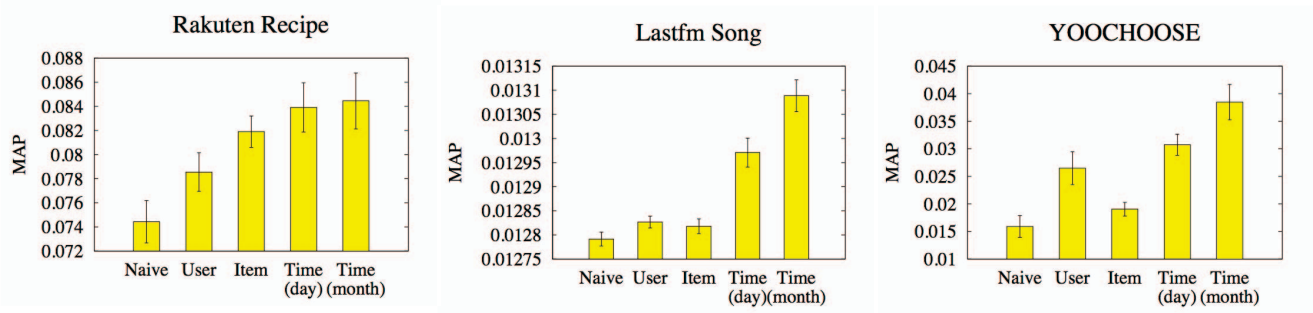

Figure 2. Experimental Results

Table IV
EXPERIMENTAL RESULTS FOR USER GROUPS DIVIDED WITH THE NUMBER OF FEEDBACKS

bin	Naive	User	Item	Time(day)	Time(month)
bin(1)	0.05617	-0.00011	+0.00006	+0.00503	**+0.01004**
bin(2)	0.07115	+0.00079	+0.00180	+0.00595	**+0.01004**
bin(3)	0.08981	+0.00130	+0.00235	+0.00544	**+0.00708**
bin(4)	0.08986	+0.00534	+0.00625	+0.01013	**+0.01242**
bin(5)	0.08169	+0.00381	+0.00494	+0.00682	**+0.00710**
bin(6)	0.06295	+0.00361	+0.00633	+0.01104	**+0.01483**
bin(7)	0.06608	+0.00637	**+0.01039**	+0.01000	+0.00950
bin(8)	0.09639	+0.00726	+0.01573	**+0.01653**	+0.01266
bin(9)	0.08412	+0.00986	**+0.02002**	+0.01868	+0.01434
bin(10)	0.04663	+0.00288	**+0.00685**	+0.00520	+0.00219

Table V
EXPERIMENTAL RESULTS FOR USER GROUPS DIVIDED WITH THE AVERAGE OF ITEM POPULARITY

bin	Naive	User	Item	Time(day)	Time(month)
bin(1)	0.11177	+0.01066	+0.01460	**+0.01859**	+0.01415
bin(2)	0.12218	+0.00609	+0.00899	**+0.01485**	+0.01306
bin(3)	0.10683	+0.00641	+0.01039	**+0.01316**	+0.01252
bin(4)	0.08343	+0.00454	+0.00845	+0.01068	**+0.01080**
bin(5)	0.06587	+0.00237	+0.00705	+0.00804	**+0.00899**
bin(6)	0.05990	+0.00371	+0.00734	+0.00796	**+0.01043**
bin(7)	0.05455	+0.00268	+0.00487	+0.00676	**+0.00920**
bin(8)	0.04910	+0.00043	+0.00193	+0.00391	**+0.00818**
bin(9)	0.04976	+0.00205	+0.00586	+0.00653	**+0.00894**
bin(10)	0.04145	+0.00219	**+0.00524**	+0.00436	+0.00393