SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

Volume 6 No. 2
ISSN 2054-7390

Expert CF: Solving Data Matrix Sparsity and Computation Complexity Problems

Gangmin Li and Minghuang Chi
Department of Computer Science and Software Engineering
Xi'an Jiaotong-Liverpool University
Gangmin.li@xjtlu.edu.cn

ABSTRACT

Collaborative Filtering (CF) is widely used to provide recommendations in ecommerce systems. CF works on a large data set by constructing an item-user matrix through association analyses among items and similarity analyses among users. However, CF suffers from data sparsity and computation complexity. This paper introduces a concept of "experts" to overcome the two identified problems. An expert is an artificially created user, who represents a cluster of users in terms of behavior and taste. The construction of experts can be done off-line through data filtering and classification. In actual recommendation, when data are spars, a number of experts can be added as existing users to predicate shopping habit for a particular user. The mechanism of "off-line expert's construction" and "on-line expert's addition" in recommendation not only to overcome the data sparsity but also improving on scalability by reduce computation in recommendation phase.

Keywords: Recommendation, Collaborative Filtering, Artificial Intelligence, Experts, Data Matrix Completion

1 Introduction

Recommendation systems developed for commerce have been one of the great successes in Big Data application [1, 2]. Recommendation systems use input data about a customer's interests to generate a list of recommendations. The key problem associated with the recommendation system is to define customer's interests. Many applications use only the items that customers purchase and explicitly rate to represent their interests, but they can also use other attributes, including items viewed, demographic data, subject interests, and favorite artists etc. It is generally belief that recommendations that are more precise can be made if available data are increasingly large. As consequences, that larger and larger consumer's data are collected. Confronting such mass data, cloud computing and big data analytics methods provide e-commerce makers an alternative to large-scale data storage and HPC facilities [4, 5]. Collaborative Filtering used in the most popular recommender systems and has proved successful [3]. It offers an idea to recommend items to user from other similar users profiled together whose are found similar taste by defining their distance [6]. Distance definition among items and users suffers from data sparsity [7, 8] and cold-start problem [10], where a new item or user's entry does not have enough data to be used to make any valid recommendations [7].

DOI: 10.14738/tmlai.62.4337
Publication Date: 16th April, 2018
URL: http://dx.doi.org/10.14738/tmlai.62.4337

2 CF and EXPERTS

Proposed expert is built based on existing Collaborative Filtering (CF).

2.1 Collaborative Filtering (CF)

CF approach formulates problem as a large-scale item-user matrix, let us denote it as X and illustrated in Figure 1.

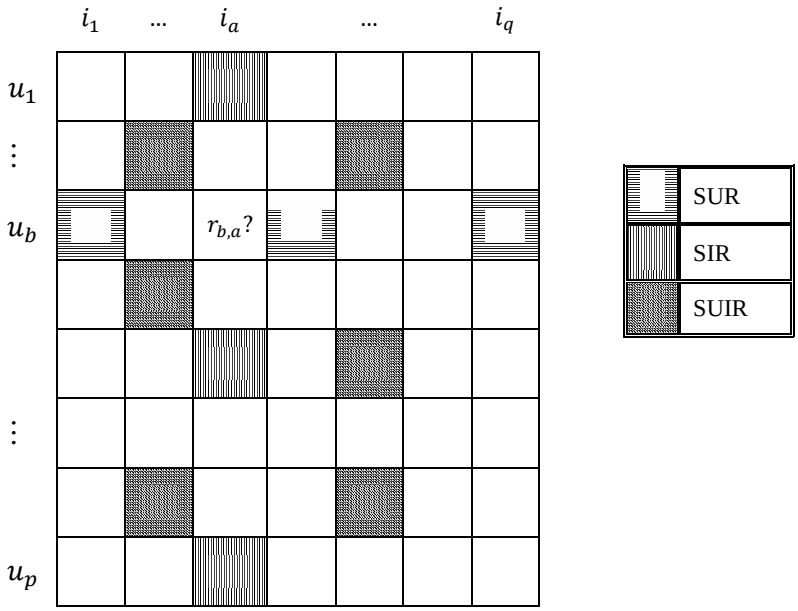

Figure 1. Traditional Collaborative Filtering Approach

As Figure 1 illustrates, user files are represented as an $Q \times P$ item-user matrix X, where Q, P are the sizes of items and users. CF intents to predict $r_{b,a}$, which is active item i_a made by active user u_b through finding other active users correlated to u_b and other active items (ratings made by active users with other existing ratings) correlated to i_a [8]. It would be easily realized when item-user matrix is filled. The two problem associated with CF are data sparsity and computation complexity.

Conversely, it is difficult to match user activities (ratings) in such enormous item set. It is also difficult to finding similarity between item sets (through item ratings) because that the rating data matrix is quite sparse. This problem becomes even serious when a user does not have much activities data at all, which is cold-start problem.

In order to formally describe the CF, we adopt a set of notations. Let

- $I = \{i_1, i_2, \ldots, i_Q\}$ and $U = \{u_1, u_2, \ldots, u_P\}$ be the sets of items and users in X,
- $\{C_u^1, C_u^2, \ldots, C_u^L\}$ be L user clusters, and users in each cluster share some similar tastes (i.e., rate similar items in a similar way),
- $I\{u\}$, $I\{C_u\}$ and $U\{i\}$ be the set of items rated by user u, the set of items rated by user cluster Cu, and the set of users who have rated item i,

- r_{u_b,i_a} denote the score that user u_b rates item i_a, $\overline{r_{i_a}}$ and $\overline{r_{u_b}}$ represent the average ratings of item i_a and user u_b,
- SI, SU and SUI be the sets of the similar items, like-minded users, and similar items and like-minded users,
- SIR, SUR and SUIR (see figure 1) denote predicting user interest over the entire item-user matrix from the ratings of the same user make on the similar items, the like-minded users make on the same item, and the like-minded users make on the similar items, i.e., SIR, SUR and SUIR predict unrated items for active users based on SI, SU and SUI, respectively.
- SR represent predicting user interest from all the ratings, i.e., SIR, SUR and SUIR,
- SIR', SUR', SUIR' and SR' be the counterparts of SIR, SUR, SUIR and SR, but they are calculated over the experts added item-user matrix X'.

Then, the item vector of the matrix X is:

$$X_i = [i_1, i_2, \dots, i_Q],\ i_q == [r_{1,q}, r_{2,q}, \dots, r_{P,q}]^T, \tag{1}$$

where q $\in [1,\ Q]$. Each column vector i_m corresponds to the ratings of a particular item *m* by *P* users. Matrix X can also be represented by user vectors illustrated as:

$$X_u = [u_1, u_2, \dots, u_P]^T, u_P = [r_{p,1}, r_{p,2}, \dots, r_{P,Q}]^T, \tag{2}$$

where $p \in [1,\ P]$. Each row vector ${u_P}^T$ indicates a user profile that represents a particular user's item ratings.

Traditional CF can provide item-based CF, user-based CF and combination of both.

Item-based CF, represented as SIR in Figure 1, finds similar items among item vectors and then use their ratings made by the same user to predict the user's rating for new item. Given an active item i_a and a user u_b, Eq. 4 denotes the mechanism of item-based CF, where $sim_{i_a,\, i_c}$ is the similarity of items i_a and i_c that can be computed by any similarity calculation. We use Pearson Correlation Coefficient (PCC) as shown in Eq 3.

$$sim_{i_a,i_b} = \frac{\sum_{u\in U}(r_{u,\,i_a} - \overline{r_{i_a}}) \cdot (r_{u,\,i_b} - \overline{r_{i_b}})}{\sqrt{\sum_{u\in U}\left(r_{u,\,i_a} - \overline{r_{i_a}}\right)^2} \cdot \sqrt{\sum_{u\in U}\left(r_{u,\,i_b} - \overline{r_{i_b}}\right)^2}} \tag{3}$$

The SIR can then be expressed in Eq. 4 as follows,

$$SIR\colon r_{u_b,i_a} = \frac{\sum_{i_c\, \epsilon\, SI} sim_{i_a,i_c} \cdot r_{u_b,i_c}}{\sum_{i_c\, \epsilon\, SI} sim_{i_a,i_c}} \tag{4}$$

User-based CF, represented as SUR in Figure 1, predicts user's rating based on the ratings of like-minded users made on the active items. Eq. 5 shows the user-based rating prediction, where sim_{u_b,u_c} is the similarity of users u_b and u_c. Similarly, a number of similarity algorithms can be used to calculate similarity between two users.

$$SUR: r_{u_b,i_a} = \frac{\sum_{u_c \epsilon SU} sim_{u_b,u_c} \cdot r_{u_c,i_a}}{\sum_{u_c \epsilon SU} sim_{u_b,u_c}}. \quad (5)$$

Above the sim_{u_a,u_b} is the similarity between user u_a and u_b. It can be obtained by using PCC as shown in Eq. 6,

$$sim_{u_a,u_b} = \frac{\sum_{i \in I}(r_{u_a,i} - \overline{r_{u_a}}) \cdot (r_{u_b,i} - \overline{r_{u_b}})}{\sqrt{\sum_{i \in I}\left(r_{u_a,i} - \overline{r_{u_a}}\right)^2} \cdot \sqrt{\sum_{i \in I}\left(r_{u_b,i} - \overline{r_{u_b}}\right)^2}} \quad (6)$$

It is obvious that search active users and items through entire item-user matrix, which is needed for both calculate item and user similarity, is computationally expensive and time consuming. Its scalability is seriously in doubt.

2.2 Expert Generation

To solve data sparsity, concept of expert opinion was introduced by Amatriain et. al. [9]. It provides extra information, which is used to predict the behavior of the general population. In our research, we solidified the expert opinion by define experts as filtered users who are qualified to act as expert in the user matrix X_u must satisfy:

1. **Rating numbers exceed a threshold ρ.** Sufficient data samples are necessary to predict the general population. Expert set needs a general coverage on the item set, that is, less sparse and more even distributed than other users in the item-user matrix. i.e.

$$u_i \in \mathrm{U} \cap \mathrm{E} \Leftrightarrow \exists\, \rho \,\&\, u_i,\ Size(I\{u_i\}) \geq \rho\ , \quad \textbf{Con}.\ 1$$

   where E is set of expert.

2. **Ratings exclude individual bias.** Expert has a fixed marking scale and deviate, provided every item he or she rates fair, and deviate less than all other user's. i.e.

$$u_i \in \mathrm{U} \cap \mathrm{E} \Leftrightarrow \sigma(r_{u_i}) < \sigma(r_U)\ , \quad \textbf{Con .2}$$

Constructed Expert based on the two conditions needs a number of steps. The first step is to cluster active users.

Clustering Users ($C_i \leftarrow U$). In this step, all users are assigned into clusters using spectral clustering. Of course, user matrix X_u is actually represented by the rating matrix (see Eq. 2). As the name suggests, spectral clustering makes use of the spectrum (it also refers to eigenvalues) of the similarity matrix of user similarity. The similarity can be obtained by using PCC on X_u. Therefore, user similarity matrix can be created. Denote $S \in R^{P \times P}$ since there are *P* users,

Here Gaussian function is used to describe similarity in previously generated similarity matrix S. Note that similarity in S is represented in a simple version S_{ab}, denote value in a's row and b's column. It is obvious that $S_{ab} = S_{ba}$ and $S_{aa} = 1$.

$$S_{ab} = \exp(-\frac{\|u_a - u_b\|}{2\sigma x^2}) \tag{7}$$

where, $\|u_a - u_b\|$, is the distance between user u_a and u_b. It can be obtained from the similarity matrix by inverting similarity into dissimilarity, which is distance matrix. σ is a scaling parameter that controls how rapidly the similarity S_{ab} reduces with the distance between u_a and u_b. Then constructing diagonal degree matrix *D* and diagonal elements *d* defined as Eq. 7:

$$d_{aa} = \sum_{b=1}^{p} S(u_a, u_b). \tag{8}$$

And calculate normalized Laplasse matrix L as Eq. 8:

$$L = I - D^{-\frac{1}{2}} S D^{-\frac{1}{2}} \tag{9}$$

Its elements are given by,

$$L_{a,b} = \begin{cases} 1 & \text{if } a = b \text{ and } d_a \neq 0 \\ -\dfrac{1}{\sqrt{d(S_{u_a}) d(S_{u_b})}} & \text{if } a \neq b \text{ and } u_a \text{ is adjacent to } u_b \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Known that *L* has *k* zero-eigenvalues that are also the *k* smallest ones. Denote $V \in R^{p \times k}$ as their corresponding eigenvectors:

$$V = [\boldsymbol{v_1}, \boldsymbol{v_2}, \dots, \boldsymbol{v_k}] = D^{\frac{1}{2}} E, \tag{11}$$

where

$$E = \begin{bmatrix} e_1 & & \\ & \ddots & \\ & & e_k \end{bmatrix},$$

where $e_i, i = 1, \dots . k$ are vectors of all ones. Thus, one needs to find the first eigenvectors of *L* (i.e., eigenvectors corresponding to the *k* smallest eigenvalues). The result is presented a single column matrix,

the rows indicates orthogonal points on the unit sphere and can be clustered by k-means. Once all users have been assigned into different clusters, we can represent each user cluster with its centroid $\bar{u}$. The centroid is the artificially generated expert. The next step is to choose required number of experts from similarity clusters.

Choose Experts. Once we have all active users clustered into different user clusters, then we would store the similarity for each user to user-cluster and it gives every user similar clusters in a descending order. These clusters are used for selecting the top *K* like-minded users and generating a much smaller matrix that can be accessed quickly.

Given an item set $I = I\{u_a\} \cap I\{C_{u'}\}$, define the similarity between user u_a and user cluster $C_{u'}$ as follows it is again PCC similarity measurement.

$$sim_{u_a,C_{u'}} = \frac{\sum_{i\in \mathrm{I}\Delta} r_{C_{u',i}} \cdot (r_{u_a,i} - \overline{r_{u_a}})}{\sqrt{\sum_{i\in \mathcal{J}} (\Delta\, r_{C_{u',i}})^2} \cdot \sqrt{\sum_{i\in \mathcal{J}} (r_{u_a,i} - \overline{r_{u_a}})^2}} \tag{12}$$

2.3 Use of Expert

In the on-line phase, requests need to be processed and responded quickly. Having experts make data matrix can be dense, but search larger space still time consuming. Therefore, a locally-reduced item-user matrix can be created and used. A locally-reduced item-user matrix selects the most similar items and the most similar active user supplemented with the most similar experts.

Creating a Locally-Reduced $\boldsymbol{M} \times \boldsymbol{K}$ Item-User Matrix. Locally-Reduced $\boldsymbol{M} \times \boldsymbol{K}$ Item-User Matrix has M top similar items and N top like-minded users and E top like-mined experts. Here N + E = K. It is relatively easy to select M top similar items SI, Following Eq. 3 similarity between two items can be calculated and all the similarity can be sorted in a descending order. The top M similar items can be selected as reduced item set for the online matrix. Similarly, the top N can be selected from a previous build link-minded user set SU. So the user set of the online matrix is also reduced. If there are E empty cells in the user set to form M × K Item-User Matrix, K experts can be selected from the similar experts set.

Note that ratings of expert and like-minded user can be in different scale from the original rating style and a weight may be needed during practical recommendation. A parameter $\boldsymbol{\omega}$ defined as:

$$\omega\colon\ \omega_{u,i} = \begin{cases} \varepsilon & if\ u\ rates\ i \\ 1-\varepsilon & otherwise \end{cases}, \tag{13}$$

After the top M similar items and K like-minded users are selected, related ratings will be extracted to fill the locally-reduced item-user matrix from original item-user matrix. So predicts user's request and makes recommendation can be made based on experts.

2.4 Predicted User Rate with Expert Added

Both like-minded user and expert's rate can be used to predict a user made on the same item. Four types of ratings can be obtained, ratings from the same user made on the similar items, like-minded user made on the same item, like-minded users made on similar items and like-minded expert made on the same item. Denote as SIR', SUR', $SUIR'$, and SER'. Given an active i_a and u_b,:

$$
\begin{aligned}
SIR' &= \frac{\sum_{s=1}^{M} \omega \cdot sim_{i_s,i_a} \cdot r_{u_b,i_s}}{\sum_{s=1}^{M} \omega \cdot sim_{i_s,i_a}} \\
SUR' &= \frac{\sum_{t=1}^{K} \omega \cdot sim_{u_t,u_b} \cdot (r_{u_t,i_a} - \overline{r_{u_t}})}{\sum_{t=1}^{K} \omega \cdot sim_{u_t,u_b}} + \overline{r_{u_b}} \\
SUIR' &= \frac{\sum_{t=1}^{K} \sum_{s=1}^{M} \omega \cdot sim_{(i_s,i_a),(u_t,u_b)} \cdot r_{u,i}}{\sum_{t=1}^{K} \sum_{s=1}^{M} \omega \cdot sim_{(i_s,i_a),(u_t,u_b)}} \\
SER' &= \frac{\sum_{t=1}^{E} \omega \cdot sim_{e_t,u_b} \cdot (r_{e_t,i_a} - \overline{r_{e_t}})}{\sum_{t=1}^{E} \omega \cdot sim_{e_t,u_b}} + \overline{r_{u_b}}
\end{aligned}
\tag{14}
$$

where $sim_{(i_s,i_a),(u_t,u_b)}$ is defined:

$$
sim_{(i_s,i_a),(u_t,u_b)} = \frac{sim_{i_s,i_a} \cdot sim_{u_t,u_b}}{\sqrt{sim^2{}_{i_s,i_a} + sim^2{}_{u_t,u_b}}} \tag{15}
$$

In practice, SIR'， SUR', $SUIR'$, and SER' can then be fused to extract the prediction rating by a fusing function. λ, δ *and* θ are three parameters introduced to balance SIR'， SUR', $SUIR'$, and SER' in order to achieve better recommendation and fit these ratings to real conditions. The function is defined:

$$
\begin{aligned}
SR': \widehat{r_{u_b,i_a}} = £\{SIR',\ SUR', SUIR', SER'\} \\
= (1-\lambda)\cdot(1-\delta)\cdot(1-\theta)\cdot SIR' \\
+(1-\lambda)\cdot(1-\delta)\cdot\theta\cdot SUR' \\
+(1-\lambda)\cdot\delta\cdot SER' \\
+\lambda\cdot SUIR'
\end{aligned}
\tag{16}
$$

3 Evaluation

An *Expert*CF is implemented in Python, a succinct and fast programming language. The evaluation is done by using MovieLens, a data set from GroupLens Research at the University of Minnesota, It is one of the most popular machine learning data sets for recommender system. Its original dataset contains 138,000 users and 27,000 movies. Our evaluation only used partial data for training and partial data for testing. The division of the dataset is in proportion (60%:40%) and 60% used for training and 40% is used for testing. The program was run with Windows 10 64-bit Operating System with 16GB RAM and 3.40GHz.

MAE metric is adopted in our tests as it is the most used metric in recommender system offline testing. The function is defined as follows:

$$
MAE = \frac{\sum_{u \in T} \left| r_{u,i} - \widehat{r_{u,i}} \right|}{|T|} \tag{17}
$$

Where $r_{u,i}$ denotes ratings that user *u* rates on item *i* and $\widehat{r_{u,i}}$ is the predicted rating. *T* denotes the test set with $|T|$ presenting the size of it. As the function illustrates the difference between real value and predicted value comes smaller, the more accurate the recommender performs, which means lower MAE indicates higher accuracy.

3.1 Overall Performances in terms of accuracy

In this evaluation, *expert*CF was compared with traditional memory-based CF approaches: an item-based approach using PCC (SIR) and a user-based approach using PCC (SUR), both also used in *expert*CF as offline calculation to improve accuracy performance.

The dataset from MovieLens includes 27,000 movies and 138,000 users. We randomly extracted 500 users each user rated over 40 movies and they were grouped by selecting the first 100, 200 and 300 users, denoted as ML_100, ML_200 and ML_300, as the training set. The last 200 users were used as the test set. We varied the number of items rated by active users from 5, 10 to 20, denoted as Given5, Given10 and Given20. Few ratings were not adding any improvement to our predictions. Therefore, sparse users were cleaned though choosing artificially generated expert cluster E_i. As the relationship between minimum rating threshold and expert numbers selected among active users illustrated in Figure 2, a final threshold for number of ratings a user has to make to be an expert was set 250. The rating scale E_i was set to 8294 to be an expert since an expert has to have a fixed marking scale and deviate, to evidently demonstrate every item he or she rates fair, and deviate less from one's to another's.

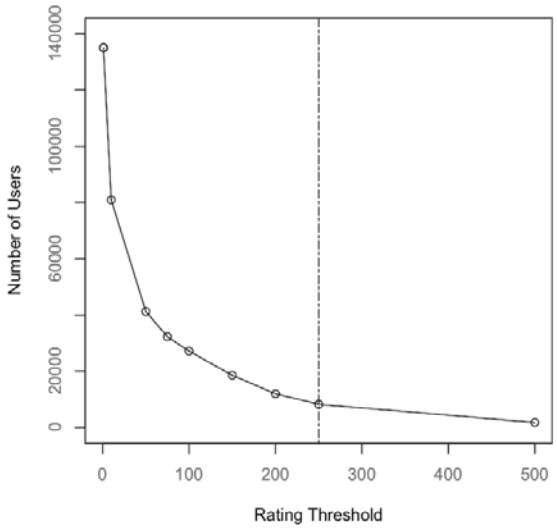

Figure 2 Relations Between Minimum Rating Threshold and Expert Numbers Selected

The other parameters of *expert*CF are previously set as follows: C=50, λ=0.7, δ=0.1, θ=0.1, σ=0.55, K=25, M=95, E=10 and ω=0.35. As **Table 2** demonstrated, *expert*CF outperforms the SUR and SIR considerably with respect to recommendation accuracy.

Table 2. MAEs on MovieLens among different CF Approaches

Training Set	Method	Given5	Given10	Given20
ML_300	*expert***CF**	0.765	0.744	0.721
	SUR	0.838	0.814	0.802
	SIR	0.870	0.838	0.813
ML_200	*expert***CF**	0.793	0.757	0.731
	SUR	0.843	0.822	0.807
	SIR	0.855	0.834	0.812
ML_100	*expert***CF**	0.802	0.779	0.770
	SUR	0.876	0.847	0.811
	SIR	0.890	0.801	0.824

3.2 Expert Cluster Coverage

In the previous study, it is found that in actual recommendation error includes a great portion of what brought by the noise in the users' explicit feedback (biased). *Expert* may not increase the accuracy rather to eliminate partial errors caused by the data sparsity. So it is interesting to see how expert can solve cold-start problem and to enhance the coverage. The second evaluation is to see how expert to make up coverage where experts typically are motivated to rate new item, it solves new item entry, in other words, new item leans to be recommended.

In this evaluation, supposed parameters stay the same, coverage rate for *expert*CF and other three state-of-the-art recommenders are given in Table 3. Where CFSF is CF only involves smooth and fusion [12]. Only ML_300 and given 20 are used.

Table 3. Coverage Rate on ML_300 for the different CF Approaches

Training Set	Method	Given20	Coverage
ML_300	*expert***SFCF**	0.721	**96.9%**
	CFSF	0.705	94.1%
	SUR	0.802	91.2%
	SIR	0.813	93.6%

The results show the coverage increase significantly in comparison with other approaches.

4 Conclusion

Expert CF build based on existing approached with aims to resolve problems of data sparsity and scalability associated with the CF in recommendation systems. This paper reports our efforts in artificially generated experts based on the existing data. Evaluations show the improvements on the solving problems. Accuracy is improved when data is sparse and coverage is also improved. However, a few deficiencies need future improvement. The original expectation intended to cover all users request, that is, to achieve 100% coverage, toward complete solution of cold-start issue. The system addressed this problem in which

provided fusing function to fuse rating prediction with a fixed weight whereas, from a consideration of original goal, a separate function that deals with identification of an inactive user or new user would effectively solve this problem.

Meanwhile, dynamic expert set depends on user activity. Supposed using matrix factorization to detect each user's eigenvalue, which the result could indicate user's interest and make experts sample become entire original user set, each user could be an expert in a specific condition and the coverage rate could reach further improvement.

REFFERENCE

[1]. J.B. Schafer, J.A. Konstan, and J. Reidl, "E-Commerce Recommendation Applications," Data Mining and Knowledge Discovery, Kluwer Academic, 2001, pp. 115-153.

[2]. Greg Linden, Brent Smith, and Jeremy York， "Amazon.com Recommendations Item-to-Item Collaborative Filtering" IEEE INTERNET COMPUTING

[3]. Gediminas Adomavicius, Alexander Tuzhilin. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*. **17**, 734-749.

[4]. F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, R. E. Gruber. (2006). Bigtable: a distributed storage system for structured data. *Proceedings of OSDI*, Berkeley, CA, USA. USENIX Association. 205-218.

[5]. WY Chen，Y Song，H Bai，CJ Lin，EY Chang. (2011). Parallel spectral clustering in distributed systems. *IEEE Transactions on Software Engineering*. **33(3)**, 568-586.

[6]. G. Linden, B. Smith, J. York. (2003). Amazon.com Recommendations: Item-to-Item Collaborative Filtering, *IEEE Internet Computing, Jan./Feb.*

[7]. Nathan Nan Liu, Xiangrui Meng, Chao Liu, Qiang Yang. (2011). Wisdom of the Better Few: Cold Start Recommendation via Representative based Rating Elicitation. Proceedings of the 5th ACM Recommender Systems Conference.

[8]. Daqiang Zhang, Jiannong Cao, Jingyu Zhou, Minyi Guo and Vaskar Raychoudhury. (2009). An Efficient Collaborative Filtering Approach Using Smoothing and Fusing. *2009 International Conference on Parallel Processing*. 558-565.

[9]. X Amatriain，N Lathia，JM Pujol，H Kwak and N Oliver. (2009). The Wisdom of the Few A Collaborative Filtering Approach Based on Expert Opinions from the Web. International Acm Sigir Conference on Research & Development in Information Retrieval. 532-539.

[10]. Gangmin Li, Minghuang Chi and Gautam Pal. (2017). Expert CF: Sparse data matrix completion with artificial experts. International Conference on Recent Advancements in Computing, IoT and Computer Engineering Technology (CICET'17). Taipei, Taiwan October 23-25, 2017.