A New Algorithm Based on Item Clustering and Matrix Factorization

Xu Wang, Xingjun Wang, Zhixiong Ding, Xinxin Nie, and Linghao Xiao

***Abstract*—As of today, the ability of providing personalized user experience has been a critical factor to determine whether a company can be successful or not. Then both academic and industry have devoted a lot of energy to promoting its development. In this paper, with the purpose of generating recommendation ranked lists, we put forward a new recommender scheme based on item clustering and matrix factorization. First, we raise a novel clustering algorithm using distance to obtain latent factors, which gathers items similar to each others successfully. Using the latent factors got from clusters, we generate the item factor vector. In addition, learned from the idea of SVD(Singular Value Decomposition), we adopt matrix factorization to finish the matrix completion. By making a comparison with other algorithms, our approach performs better.**

***Index Terms*—Recommender systems, clustering algorithm, matrix factorization, latent factor.**

I. Introduction

Nowadays, the ability of providing personalized user experience has been a critical factor to determine whether a company can be successful or not. This trend is led by e-commerce firms like Amazon, then extended to almost every category. At present, personalized user recommendations are generally offered by news webs, music and video sharing platforms, and social networks and etc.

In the content of recommender systems , the data consists of the items, the users and the feedback that users have done to the items. Recommender systems are to generate a limited list of items that as far as possible to meet users' tastes. Collaborative filtering is a technique, utilizing user behaviors, commonly applied in real recommender systems. And MF, which is short for Matrix Factorization, is one of the most popular methods for collaborative filtering. In the MF model, user and item are quantized by latent factor vector.

Clustering is an efficient approach to get latent factors. In [1], YS Cho et al. propose a new clustering algorithm using the weighted preference based on recency, frequency and monetary score for personalized recommendation in u-commerce under ubiquitous computing environment which is required by real time accessibility and agility. A new collaborative filtering algorithm based on clustering is put forward in [2]. Based on users' ratings on items, it applies K-means clustering algorithm to cluster items into classes and calculates the similarity of users in each cluster. They also introduce the factor of overlap to optimize the accuracy of the local similarity between users.

MF is one of the most effective approaches applied in recommender systems. In MF models, both calculating users' ratings on items and sorting items are time-consuming processes, Petroni F. *et al.* [3] discuss a distributed asynchronous variant of stochastic gradient descent to improve the computational efficiency of the algorithm. Yoram Bachrach *et al.*[4] propose a novel order preserving transformation, mapping the maximum inner product search problem to Euclidean space nearest neighbor search problem, which also upgrades the efficiency. Some people are try to introduce other information into MF models. Chen Cheng *et al.* [5] combine Factorization Machines with auxiliary information available and put forward a novel Gradient Boosting Factorization Machine model to incorporate feature selection algorithm with Factorization Machines into a unified framework. Temporal influence [6], ratings clustering [7] and explicit trust in society [8] are also introduced to improve the accuracy of MF models.

In the following sections, we will introduce some related works about our approach in section II. Then we will give a detail description of our recommender scheme, mainly focuses on presenting our item clustering algorithm and how to build user and item factor vectors in section III. In section IV, we discuss the experiment results and make a comparison with other recommender algorithms. Finally, we make a summary about our work, then argue what to do next.

II. Related Works

A. Collaborative Filtering

If a algorithm makes personalized user experience based on users' historical behaviors, then it can be called collaborative filtering(CF). CF is processed in user-item rating matrix. The matrix is expressed as $m \times n$ rating matrix R_{mn}. The m is number of users and n is number of items. The r_{ui} in R_{mn} represents the preference of user u on item i.

$$R_{mn} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \cdots & \cdots & \ddots & \cdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix} \tag{1}$$

Manuscript received December 1, 2015; revised March 1, 2016.

Xu Wang, Xingjun Wang, and Zhixing Ding are with the Electrical Engineering Department, Tsinghua University, China (e-mail: wangxu13@mails.tsinghua.edu.cn, wangxingjun@tsinghua.edu.cn, dzx13@mails.tsinghua.edu.cn).

Xinxin Nie and Linghao Xiao are with the Computer Science Department, University of East Anglia, UK (e-mail: xinxin.nie@uea.ac.uk, xlh14@mails.tsinghua.edu.cn).

There are two steps to do to finish the collaborative filtering. Take the user-based CF for example, the first and critical step is to compute the similarity between users and to select the nearest neighbors of users. There are a lot of methods to compute the similarity, most popular are euclidean distance-based similarity, mahalanobis distance, cosine-based similarity and pearson correlation coefficient.

The second step is to get the prediction. Once we get the neighbors of target user, then we have to use an technique to get the prediction of target user u to item i. Here is an approach to compute the value of prediction:

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in N_u} similarity(u,v)(r_{vi} - \bar{r}_v)}{\sum_{v \in N_u} similarity(u,v)} \tag{2}$$

where N_u is the top-N nearest neighbors of user u, and $\bar{r}_u$ is the average rating of user u.

Though CF has been widely used in real recommender systems, there still exists a tough problem: cold-start. Cold-start is the problem that when a new user logs up the system, what items are going to recommend to him, or when a new item enters into system, what groups of users that we recommend it to. A lot of people devote themselves to solve this problem. In [9], Sedhain S *et al*. introduce social information, like relationship in Facebook, into CF recommender system. They propose a novel social CF framework that generalizes standard item-based CF to solve the cold-start problem. When a new user comes, their recommend items to the user based on popular items, demographics, and the social friend network. In [10] , Trevisiol *et al*. take full advantage of browsing traces to solve the problem of cold-start for news recommendation. Based on browsing traces, they build a BrowseGraph, and define the ReferrerGraph as its subgraph induced by the sessions with the same referrer domain.

B. *Clustering Algorithms*

Clustering, also called unsupervised learning, is to allocate an item into a group, so that items in the same group are much more similar than those in different groups. The purpose of clustering is to find the meaningful groups hidden in datasets. There are two main categories of clustering algorithms: hierarchical and partitional[11]. Hierarchical clustering algorithms cluster items within found clusters. Partitional clustering algorithms successfully divide data into none-overlapping clusters so that each item can not be in more than one cluster.

Actually, clustering algorithms have been widely used in recommendation system. Xue *et al*. [12] raised an typical clustering algorithm applied to recommendation system, They used k-means algorithm to help to build the neighborhood. They do not restrict neighborhood to the cluster that user belongs to but rather use the distance from user to different cluster centers as a pre-selection step for the neighbors. The smooth technology based on clustering is achieved by them, and the result as reported is better than knn-based cf. Similarly, Sarwar *et al*.[13] present an method to implement a extendible kNN classifier. They apply the bisecting k-means algorithm to divide the user space and based on these clusters to format the neighborhood. They report that the accuracy rate has decreased about 5%, but the efficiency has been greatly improved. Connor and Herlocker put forward another method, clustering items not users. Using the Pearson Correlation similarity measure, they try four different algorithms: average link hierarchical agglomerative[14], robust clustering algorithm for categorical attributes, kMetis and hMetis. Even though the efficiency is improved, the accuracy rate and coverage has got bad. At last, Li *et al*.[15] , Ungar and Foster[16] come up with a similar idea , using k-means clustering to solve a probabilistic model interpretation of the recommender problem.

C. *Matrix Factorization*

In MF models, each user u and item i are described as latent vector $q_u, p_i \in R^d$. The predicted preference of user u to item i is computed according to the rule:

$$\hat{r}_{ui} = overrallMean + b_u + b_i + p_i q_u^T \tag{3}$$

where $overrallMean$ is the average rating of user u, and b_u and b_i represent user u and item i biases respectively. This model can be easily extended to different kinds of user feedback.

Though $overrallMean$ and b_u are very crucial parts of the rule, but we can see that they don't determine the ranking of items in recommendation list. Hence, the rule can be rewrite as $\hat{r}_{ui} = b_i + p_i q_u^T$, which can output the same recommendation list as (2).

III. Recommendation Problems

A key contribution of our work is to come up with a novel clustering algorithm to obtain latent factors, and build the predicted rating matrix based on matrix factorization. In this section, we will make a detail description about our approach. The experimental data we use is MovieLens-100k dataset.

A. *Computing Distance*

The clustering algorithm, based on distance, is to cluster items. So the first step is compute distance between items. As mentioned in section II, we use (4)

$$dis(i,j) = \sum_{u \in U} \left| r_{ui} - r_{uj} \right| \tag{4}$$

To calculate the distance between item i and item j. The value of distance represents the similarity. We define that the smaller the distance between i and j is, the more similar they are.

B. *Clustering Algorithm*

Since we already get distances between items, then we are going to cluster items. The algorithm is as follows:

Algorithm 1: Item Clustering Algorithm

Input:

The minimum number of items in a cluster MIN_p ;

The maximum number of clusters K;

Output:

The centers of clusters $\{c_i\}$;

The result of clusters $\{C_i\}$.

Method:

1. Select two items with largest distance from training set as the initial two cluster centers $\{c_1, c_2\}$;
2. Cluster all items into two clusters based on the nearest neighbors rule;
3. Find the cluster with most items $C_{\max}$, and select the item that is the farthest from the center of $C_{\max}$ as a new clustering center;
4. Re-cluster the clusters according to nearest neighbors rules;
5. If cluster number is K, go to step 6; else, return to step 3;
6. If the number of items in a cluster is smaller than MIN_p , then put the items into set T and delete this cluster. Traverse all clusters.
7. According to the nearest neighbors rule, put items which in T into the nearest cluster.
8. Return centers $\{c_1, c_2 \cdots, c_D\}$ and clusters $\{C_1, C_2 \cdots, C_D\}$

The rule of clustering is nearest neighbors. The pivotal content of this clustering algorithm is to find new clustering centers. The approach that we apply is to seek a new center in the cluster $C_{\max}$ with most items. The item farthest from the center of $C_{\max}$, is the new center. We can see that the cluster with most items also contains cold items. The cold items mean that the number of users rating on those is small. In fact, some new interest points commonly hide in these items. So we dig the items successfully, thus extending the diversity of recommendation, which is also a evaluating indicator. So our clustering algorithm can also be seen as a process to mine new interest points.

C. Building Item Vector

In this part, we will map items to factor space. The clusters are treated as latent factors. And an item is described as a latent factor vector $p_i \in R^D$, where D is the number of clusters. Here is how we build the item vector p_i . First, we compute the distance between item i and item j using (4).

The degree of item i contains factor k is calculated according to (5):

$$\hat{p}_{ik} = \left(\sum_{j=1}^{K} \frac{d_{ik}}{d_{ij}} \right)^{-1} \tag{5}$$

After getting the $\hat{p}_{ik}$, we normalize it by

$$p_{ik} = \frac{\hat{p}_{ik}}{\sum_{d=1}^{D} \hat{p}_{id}} \tag{6}$$

Finally p_{ik} represents the degree of item i contains factor k . So the latent factor vector of item i is like this:

$$p_i = \left(p_{i1}, p_{i2}, \cdots, p_{iD}\right) \tag{7}$$

D. Generating Predicted Rating Matrix

We draw lessons from the thought of SVD(Singular Value Decomposition) to obtain the predicted rating matrix. As we know, generating personalized user recommendations can be abstracted as a matrix completion problem. We use gradient descent method to fill the rating matrix. The predicted preference or rating r_{ui} that user u to item i is computed by (3). As mentioned in section II, (3) can be simplified as

$$\hat{r}_{ui} = b_i + p_i q_u^T \tag{8}$$

In order to learn all parameters, we minimize normalized squared error:

$$\min_{b,q} \sum_{u,i} (r_{ui} - b_i - p_i q_u^T)^2 + \lambda (b_i^2 + \|p_i\|^2 + \|q_u\|^2) \tag{9}$$

The relative prediction error is defined as $e_{ui} = r_{ui} - \hat{r}_{ui}$. Then we update and correct parameters like this:

$$\begin{aligned} b_i &\leftarrow b_i + \gamma \cdot (e_{ui} - \lambda \cdot b_i) \\ q_u &\leftarrow q_u + \gamma \cdot (e_{ui} \cdot p_i - \lambda \cdot q_u) \end{aligned} \tag{10}$$

In this way, we finish the matrix completion. Each row in matrix is the quantitative representation of user. Take user u for example, it is described as $(r_{u1}, r_{u2}, \cdots, r_{ui}, \cdots, r_{uD})$, where i is index of items. We sort r_{ui} by descending order, and recommend items with high preference r_{ui} to user u .

IV. Evaluation through a Top-N Recommender

In this section, we adopt top-N recommender to evaluate the performance of our algorithm, and the experiment data we use is MovieLens-100k dataset. The top-N recommender is a practical strategy for pragmatic recommender systems. The idea of it is to recommend N items, which have not been watched by user.

A. Different Clusters

In this part, we discuss that how the number of factors or clusters affects the performance of our model.

The evaluation metrics we use are precision and recall in recommender systems. Suppose that N items are recommended to user u , which is described as $R(u)$. And the viewing set is described as $T(u)$. The intersection $R(u) \cap T(u)$ represents the correctly recommender items set. So the precision and recall are defined as follows:

$$precision = \frac{\sum_{u} |R(u) \cap T(u)|}{\sum_{u} |R(u)|} \tag{11}$$

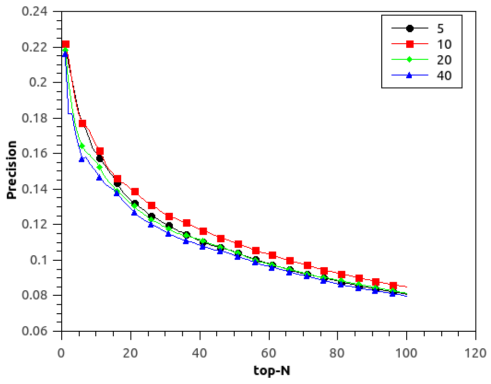

Fig. 1. The precision of our model.

$$recall = \frac{\sum_{u} |R(u) \cap T(u)|}{\sum_{u} |T(u)|} \tag{12}$$

Precision measures that how many errors we make in classifying items as being of users' favorite ones. On the other hand, recall is a measure of how good we are in not leaving out items that should have been classified as belonging to the recommendation list[11]. The number of clusters influences the value of precision and recall. The results have been shown in Fig. 1 and Fig. 2.

Fig. 1 and Fig. 2 both denote that the number of factors or clusters affects the performance of recommender system. Fig. 1 and Fig. 2 show that when top-N is 10, the precision and recall is the best. It is not true that the larger the number is, the better the results are. From Fig. 1 and Fig. 2, we can see that when the number of clusters is 20 or 40, then precision and recall almost have the same value.

In addition, Fig. 1 shows that precision is negatively related to top-N, Fig. 2 shows exactly the opposite, recall is positively related to top-N. And it is obvious that there is a trade-off between precision and recall. Then we have to ask how to decide weather the recommender system is good or not. To solve this problem, we introduce another indicator F-Measure, which is the weighted harmonic mean of precision and recall.

$$F = \frac{(a^2+1)\, precision \times recall}{a^2 (precision + recall)} \tag{13}$$

When $a=1$, it becomes commonly known $F1$, which is the indicator we put to use:

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{14}$$

It is easy to see that $F1$ takes both precision and recall into consideration. The value of $F1$ denotes the performance of recommender system. The larger the value is, the better the recommender system is. It can say that when the value of $F1$ is large enough, then the experiment result is good. The experimental results about $F1$ is shown in Fig. 3. From Fig. 3, we can see that when top-N is bigger than 35, the $F1$ almost tends to maintain constant value. In reality, there exists a threshold of top-N. If the value of top-N is bigger than its threshold, the performance of recommender systems wouldn't get better, as is shown in Fig. 3. Obviously, the threshold is not constant, it has a great relationship with dataset.

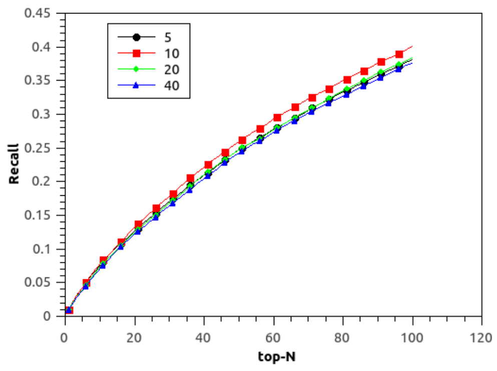

Fig. 2. The recall of our model.

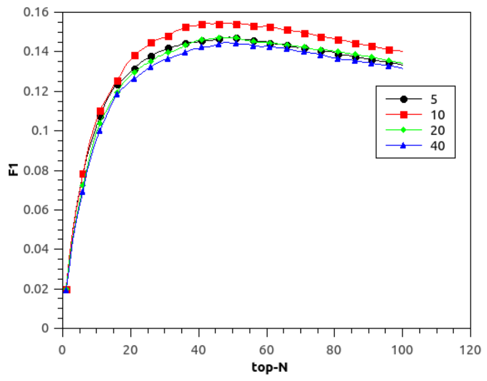

Fig. 3. The F1-measure of our model.

B. *Compare with Other Models*

We compare our approach with user-based and item-based collaborative filtering, SVD model. The results have been shown in Fig. 4, Fig. 5 and Fig. 6. For user-based collaborative filtering, the nearest neighbors N is 15, the similarity is calculated according to pearson correlation coefficient. For item-based CF, the similarity is computed according to mahalanobis distance. For SVD, the learning rate γ is 0.005 and regularization factor λ is 0.02.

From Fig. 4, Fig. 5 and Fig. 6 we can see that our model is better than other models. There is also another evaluation criterion used in Netflix Prize, that is RMSE(Root-Mean-Square Error). RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{|\Gamma|} \sum_{(u,i)\in\Gamma} (\hat{r}_{ui} - r_{ui})^2} \tag{15}$$

where Γ is testing set. As for the term of RMSE, it actually only indicates the degree of accuracy of predicted rating matrix, but not reflects weather the recommendation is good or not. However, precision and recall directly denote users' preference, since recommender systems show a list of recommended items not predicted accuracy.

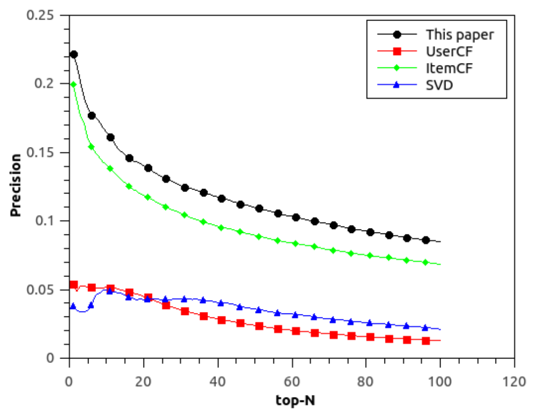

Fig. 4. The precision of models.

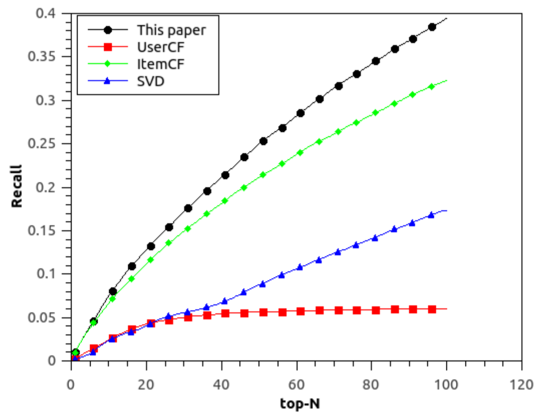

Fig. 5. The recall of models.

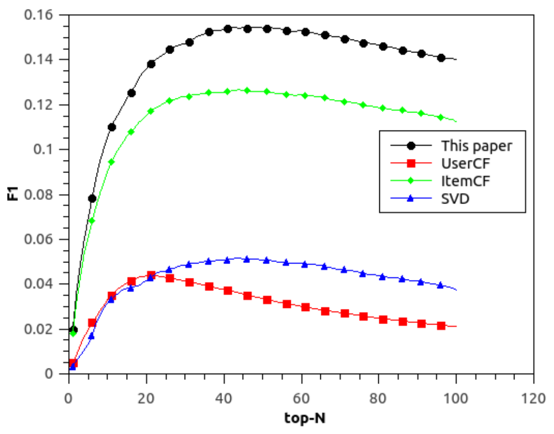

Fig. 6. The F1-measure of models.

TABLE I: Comparison of Different Models on RMSE and F1 at Top-N is 35

Method	RMSE	F1
This paper	0.724	0.154
User-based CF	1.117	0.048
Item-based CF	1.022	0.125
SVD	1.120	0.051

Table I shows evaluating indicator RMSE and $F1$ of our approach, user-based CF, item-based CF and SVD. The value of top-N is 35. It can be seen from the table that our model works better on indicators, which greatly benefits from that our clustering algorithm focuses much on finding the latent factors totally based on objective users' historical behavior, thus finding the interest points hide in the training set. And using matrix factorization can successfully quantify user and item. The experiment result demonstrates that our model works very well.

V. Conclusion

In this paper, we proposed a novel recommender structure to make top-N recommendation. First, the key part is that we propose a new clustering algorithm based on distance, which successfully dig out users' interest points. Owing to items clustering, we can obtain the latent factors. Furthermore, according to the degree of item contains factor, we generate the item vector. By matrix factorization and gradient descent method , we accomplish the matrix completion. Finally we empirically evaluated it on the MovieLens dataset. Our analysis denotes that our model allows achieving excellent quality recommendations.

Nevertheless, a lot of works still can be done to improve the performance of our approach. First, we will devote ourselves to build an model to reflect the real preference that user on item. In addition, we can still introduce some other methods to compute distance to speed up the clustering algorithm, which is a time-consuming process. There may be another way to map item and user to latent factor vectors, how to make the most of latent factors, we still have a lot to do. Last but not the least, it is a hard task to solve the cold-start problem in our model.

Acknowledgment

First, I would like to say thanks to my tutor: Xingjun Wang. In the process of composing this paper, he gives me many academic and constructive advices, and helps me to correct my paper.

At last, I am very grateful of my dear friends, Zhixiong Ding, Xinxin Nie and Linghao Xiao, who give me a lot of help and discuss with me about my experiment and paper.

References

[1] C. Y. Sung *et al.*, "Clustering method using weighted preference based on RFM score for personalized recommendation system in u-commerce," *Ubiquitous Information Technologies and Applications*, pp. 131-140.Springer Berlin Heidelberg, 2014.

[2] W. Suyun *et al.*, "Collaborative filtering recommendation algorithm based on item clustering and global similarity," in *Proc. 2012 IEE International Conference on Business Intelligence and Financial Engineering (BIFE),* pp. 69-72, 2012.

[3] P. Fabio and L. Querzoni, "GASGD: stochastic gradient descent for distributed asynchronous matrix completion via graph partitioning," in *Proc. the 8th ACM Conference on Recommender Systems,* pp. 241-248, ACM, 2014.

[4] B. Yoram *et al.*, "Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces," in *Proc. 8th ACM Conference on Recommender Systems,* pp. 257-264, ACM, 2014.

[5] C. Chen *et al.*, "Gradient boosting factorization machines," in *Proc. the 8th ACM Conference on Recommender Systems,* pp. 265-272, ACM, 2014.

[6] P. Róbert *et al.*, "Exploiting temporal influence in online recommendation," in *Proc. the 8th ACM Conference on Recommender Systems,* pp. 273-280, ACM, 2014.

[7] L. Babak *et al.*, "'Free lunch'enhancement for collaborative filtering with factorization machines," in *Proc. the 8th ACM Conference on Recommender systems,* pp. 281-284, ACM, 2014.

[8] F. Soude *et al.*, "Implicit vs. explicit trust in social matrix factorization," in *Proc. the 8th ACM Conference on Recommender Systems,* pp. 317-320, ACM, 2014..

[9] S. Suvash *et al.*, “Social collaborative filtering for cold-start recommendations,” in *Proc. the 8th ACM Conference on Recommender Systems,* pp. 345-348, ACM, 2014.

[10] T. Michele *et al*., “Cold-start news recommendation with domain-dependent browse graph,” in *Proc. 8th ACM Conference on Recommender systems,* pp. 81-88, ACM, 2014.

[11] K. B. Paul *et al*., *Recommender Systems Handbook,* Springer, 2011.

[12] X. Gui-Rong *et al*., “Scalable collaborative filtering using cluster-based smoothing,” in *Proc. the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* pp. 114-121, ACM, 2005.

[13] S. M. Badrul *et al*., “Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering,” in *Proc. the Fifth International Conference on Computer and Information Technology,* vol. 1. 2002.

[14] G. Earl, "Pattern recognition and image analysis," 1997.

[15] L. Qing and B. M. Kim, “Clustering approach for hybrid recommender system,” in *Proc. International Conference on Web Intelligence*. pp. 33-38, 2003.

[16] U. H. Lyle and D. P. Foster, “Clustering methods for collaborative filtering,” *AAAI Workshop on Recommendation Systems*. pp. 114-129, vol. 1. 1998.

Xu Wang was born in Hebei, China, in 1990. He received the B.S. degree in information engineering from Zhejing University, Zhejiang, China, in 2013. He is currently pursuing the M.S. degree in electronic and communication engineering from Tsinghua University, Beijing, China.

Since 2013, he has been a research assistant with Shenzhen Key Lab of Information Security and Digital Content Protection Technology, Graduate School at Shenzhen, Tsinghua University. His research interests include recommender system, machine learning and data mining.

Xingjun Wang was born in Liaoning, China, in 1962. He received the M.S. degree in communication and electronic system and the Ph.D degree in information and signal processing from Tsinghua University, Beijing, China, in 1991 and 1994, respectively.

From 1994 to 1996, he was a postdoctoral researcher with Deparment of Electrical Engineering, University of Michigan-Dearborn. From 1996 to 1997, he was a senior software analyst with AT&T Corporation, Canada. From 1997 to 2000, he was first a system software designer with Department of Optical Communication and then a senior system architect with Department of Broadband Wireless Access, Nortel Networks Corporation, Canada. From 2000 to 2003, he was a vice president with Legend Silicon Corporation, USA. Since 2003, he has been a research fellow with Department of Electronic Engineering, Tsinghua University, Beijing, China. He is currently the director of Shenzhen Key Lab of Information Security and Digital Content Protection Technologies. He has published over 40 peer-reviewed journal and conference papers. He holds three PCT and over 15 Chinese patents. His research interests include information security, digital content protection, video recommendation system, hybrid network and TV white space.

Dr. Wang has many awards, including, most recently, the Recruitment Program of Global Experts Professorship at Tsinghua University from the Ministry of Education of China and Guangdong 100 Elites Fellowship.

Zhixiong Ding was born in Hunan, China, in 1990. He received the B.S. degree in electronic information science and technology from University of Science and Technology of China, Hefei, China, in 2013. He is currently pursuing the M.S. degree in information and communication engineering from Tsinghua University, Beijing, China.

Since 2013, he has been a research assistant with Shenzhen Key Lab of Information Security and Digital Content Protection Technologies, Graduate School at Shenzhen, Tsinghua University. His research interests include broadband mobile wireless network, video delivery system, hybrid network, and TV white space.

Xinxin Nie was born in Hebei, China, in 1990. She received the B.S. degree in digital media from Communication University of China, Beijing, China, in 2014. She is currently pursuing the M.S. degree in computer science from University of East Anglia, Norwich, Britain.

Since 2015, she has been a research assistant with machine learning and statistics group, Graduate School at Norwich, University of East Anglia. Her research interests include recommender system, machine learning and ensembles.

Linghao Xiao was born in Hunan, China, in 1991. He received the B.S. degree in electronic and information engineering from Xidian University, Xi’an, China, in 2014. He is currently pursuing the M.S. degree in information and communication engineering from Tsinghua University, Beijing, China.

Since 2014, he has been a research assistant with Shenzhen Key Lab of Information Security and Digital Content Protection Technologies, Graduate School at Shenzhen, Tsinghua University. His research interests include hybrid network, video delivery system and TV white space.