International Journal of Database Theory and Application
Vol.9, No.8 (2016), pp. 49-60
http://dx.doi.org/10.14257/ijdta.2016.9.8.06

Research on Improved Collaborative Filtering Algorithm Based On HADOOP

Jingxia Guo[1] and Jinniu Bai [2,*]

[1,2]*Bao Tou Medical College, BaoTou 014060, china*
[1]*Guojing7223113@163.com and 2baijinniu@163.com*

Abstract

Collaborative filtering algorithm is the most used items recommendation algorithm. We find the k neighbors with the highest similarity by calculating user similarity and recommend items for users by the score of the neighbors of the items. In the paper, we propose a hybrid recommendation algorithm based on user similarity and attribute weights to solve user ratings sparsity. We obtained the weights of users like properties through learning user ratings records and combined with the user similarity for users to recommend item. Finally, we transplant the algorithm to HADOOP platform. Through the experiment, the improved collaborative filtering algorithm is better than the original algorithm in precision and parallel attribute.

Keywords*: HADOOP, MapReduce, Data Mining, Collaborative Filtering*

1. Introduction

With advancement of Internet, traditional data mining algorithm can't meet needs of information discovery of massive data. Here the traditional method is improved with the use of most recent cloud computing technology and its parallel processing ability is enhanced by using HADOOP platform [1-2]. To improve collaborative filtering algorithm and address its scoring sparsity, a score predication algorithm based on attribute weights is proposed, in the hope to reduce user rating sparseness. Then it is improved in terms of parallelized processing capability. The method includes offline and online parts. Offline part refers to data combing and similarity calculation; online part involves item recommendation to users. With the help of previous achievements, we train users' previous rating to predict their preference for commodity attribute and obtain the weight of each kind of preference [3-5]. For commodities which are not evaluated yet, we foresee users' marks with attribute weights. Scalability means making parallel improvement of the algorithm's concurrent parts and letting it run on HADOOP platform [6-8].

2. Steps of User-Based Collaborative Filtering Algorithm

User-based collaborative filtering algorithm recommends items which are not visited by one user in these four steps:

(1) Gathering of user information; convert user rating into $m \times n$ matrix;

(2) Calculate similarity between the user and others;

(3) Rank other users according to similarity from big to small; find out k users with biggest similarity and regard them as the nearest neighbors of the recommended;

(4) Utilize similarity of k nearest neighbors and their rating values of the item to predict the user's evaluation value of the item.

Algorithm pseudo code is shown in Table1.

ISSN: 2005-4270 IJDTA

Table 1. User Based Collaborative Filtering Recommendation Algorithm Pseudo Code

Input:
User - item rating table R, similar the number K, the current user u
Output:
User u recommended item p_u
Procedure : User_Based_CF (R, k, u)
1: begin
2: for (i =1;i <m;i++) do
3: if (u $\neq$ i) then
4: $W_{u,i}$ =sim (u, i)
5: end if
6: end for
7: $\text{neighbor}_u \leftarrow \varnothing$
8: $\text{neighbor}_u \leftarrow top\_k\, l\arg est\ in\ W_{u,*}$
9: $p_u \leftarrow predict(R, neighbor_u)$
10: end
11: return p_u

2.1. Expression of User Information

Construct user-item rating matrix $R(m,n)$ to express user's scoring of one item, where m implies the number of user; n is the number of item; R_{ij} means user i's scoring of item j. Figure 1 shows a simple user rating matrix, in which each line represents user's rating of item; each column indicates rating of the same item by different users; the value of unrated item is default 0; user U_1's rating of item I_1 is 4.

	I_1	I_2	I_3	I_4
U_1	4	?	5	5
U_2	4	2	1	
U_3	3	0	2	4
U_4	4	4	0	0
U_5	2	1	3	5

Figure 1. User Rating Matrix

2.2. Calculation of Similarity

In the user-based collaborative filtering algorithm, it's a critical step to calculate similarity between users or items. Generally there're following three ways to compute user similarity.

2.2.1. Cosine Similarity

In user-item matrix R, user is expressed as one-dimensional vector; the similarity between users is represented as cosine value of the two vectors; set cosine value [0,1], suggesting that the bigger the cosine value is, the higher the similarity reaches. It is shown in Formula1.

$$w_{i,j} = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \bullet \vec{j}}{\| \vec{i} \| * \| \vec{j} \|} \tag{1}$$

2.2.2. Modified Cosine Correlation

Cosine correlation doesn't consider differences of rating by different users; modified cosine correlation overcomes the shortcoming by subtracting user rating means. It is shown in Formula2.

$$w_{u,v} = \frac{\sum_{i \in I_{uv}} (r_{u,i} - \vec{r}_u)(r_{v,i} - \vec{r}_v)}{\sqrt{\sum_{i \in I_u} (r_{u,i} - \vec{r}_u)^2} \sqrt{\sum_{i \in I_v} (r_{v,i} - \vec{r}_v)^2}} \tag{2}$$

2.2.3. Pearson Similarity

Cosine correlation doesn't take into account differences of rating by different users; modified cosine correlation avoids the defect by deducting user rating means; Pearson similarity considers the rating mean values of common items by users. It is shown in Formula3.

$$w_{u,v} = \frac{\sum_{i \in I_{uv}} (r_{u,i} - \vec{r}_u)(r_{v,i} - \vec{r}_v)}{\sqrt{\sum_{i \in I_{u,v}} (r_{u,i} - \vec{r}_u)^2} \sqrt{\sum_{i \in I_{u,v}} (r_{v,i} - \vec{r}_v)^2}} \tag{3}$$

3. Prediction Recommendation Algorithm Based on Attribute Weight

In traditional collaborative filtering algorithm, it considers only item rating, without user preference. Each person is single individual, showing different fondness of different articles. From the perspective of user preferences, we recommend items to users.

In the recommendation system, items have certain description. Each item has some attributes. For example MovieLens divide movies to 19 classes, like action, comedy, war. Each movie belongs to a few classes. Depicted in abstract computer language, each item X_i is composed of n attributes, $X_i = \{A_1^i, A_2^I, A_3^i, ...A_n^i\}$. Thus item-attribute matrix S is built. It is shown in Formula4.

$$S = \begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1n} \\ s_{21} & s_{22} & \cdots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \cdots & s_{nn} \end{pmatrix} \tag{4}$$

After item-attribute matrix is created, user preference of each attribute can be mined by training. Before calculation of user u's preference of each attribute, initialize the weight $\{w_{u1}, w_{u2}, ...w_{uk}\}$ of user u's k attributes to 1. For item j, by means of the product of attribute value and its attribute weight, p_{uj} can be foreseen. Compare p_{uj} and real rating r_{uj}; if $p_{uj} > r_{uj}$ and absolute difference is bigger than preset threshold θ, it means item j's attribute value is much big in the weight list, so it divides by 2; if $p_{uj} < r_{uj}$, and absolute difference is bigger than preset threshold θ, meaning that item j's attribute value is too small in the weight list, hence it multiplies by 2. It is shown in Formula5.

$$p_{u,j} = \sum_{i=1}^{k} w_{ui} s_{ji} \tag{5}$$

After several trainings, attribute weight values of different items are acquired. With formula 5, we can predict rating of user unrated items. The weight value relates with times of training. More trainings indicate that weight value is closer to user's real preference. The main steps of the algorithm are shown in Table2.

Table 2. Predictive Recommendation Algorithm of Attribute Weights

Input:
Item attribute table S[M][N], user u rating record R[m], item K
Output:
Prediction rating p_{uk}
Procedure : BWAPredict (S, R, k)
1: $w[N] \leftarrow 1$
2: while(user u rated for item j) do
3: $pre = \sum_{i=1}^{N} S[j][i] \times w[i]$
4: $(pre < r_{uj} \&\& | pre - r_{uj} | > \theta)$
5: for (i =0;i <N;i++)
6: if (S[j][i] ==1)
7: w[i] =w[i]*2
8: end if
9: end for
10: end if
11: end if
12: end while
13: $p_{uk} = \sum_{i=1}^{N} S[k][i] \times w[i]$
14: return p_{uk}

4. Combination Recommendation Algorithm Based on User Similarity and Attribute Value Prediction

In the above part, user preference of item attribute is got through training; and based on preference, rating value of unrated items is calculated. In this paper, we combine attribute value prediction algorithm and traditional collaborative filtering algorithm. Let I_i the item set of user i's scores, and I_j the item set of user j's scores; U_{ij} means union of rated items by user i and j. Therefore we have:

$$U_{ij} = I_i \cup I_j$$

With attribute-based weight value prediction algorithm, we calculate items which are not rated but belong to U_{ij} of user i and j. Through item rating prediction, the number of user commonly rated items is increased to decrease sparseness of rating items. At last, recommendation is made to user according to user similarity.

Algorithm pseudo code is shown in Table3.

Table 3. Hybrid Recommendation Algorithm

Input:
R represents the user - item matrix, K represents the number of the most similar neighbors, u indicates the current user needs to recommend, the item attribute table S[M][N]
Output:

Recommended list of users p_u
Procedure : Combination_CF(R,k,u,S)
1: for (i =1;i<m;i++) do
2: if (u≠ i) then
3: $U_{ui} = I_u \cup I_i$
4: $N_u = U_{ui} - I_u$
5: $N_i = U_{ui} - I_i$
6: while(each item j in N_u)
7: r_{ui} = BWAPredict(s,r,j)
8: end while
9: end if
10: end for
11: $\text{neighbor}_u \leftarrow \varnothing$
12: $\text{neighbor}_u \leftarrow top\_k\, l\arg est\ in\ W_{u,*}$
13: $p_u \leftarrow predict(R, neighbor_u)$

5. Parallel Improvement

Collaborative filtering can reduce the complexity of rating prediction with some algorithms, which, however, becomes helpless in the face of enormous data. In the paper, we enhance the processing capability of collaborative filtering algorithm by making parallel improvement of it on HADOOP platform.

As observed from the above introduction of HADOOP platform, HADOOP has very powerful processing ability. It only needs to implement Map and Reduce class when writing parallelized codes. By focusing on collaborative filtering algorithm, we analyze the algorithm workflow. Traditional collaborative filtering algorithm is decomposed into

four concurrent Job tasks: data combing, nearest neighbor calculation, rating prediction and attribute weight prediction. There's order of priority between Jobs but each Job is an independent task. In certain time range, user interest can be thought unchangeable. In a short period, a few new added rating records don't affect the calculation of similar neighbors. Hence, it's no need to implement every time data combing, similarity calculation and attribute weight prediction.

5.1. Data Combing

For massive data, it's impossible to store all data in memory. Data combing in prophase refers to classifying a pile of messy data as per user ID and screening out futile information, which is helpful to data treatment in later stage. Being no correlation between data, it's possible to assign data to different execution nodes to increase the efficiency.

According to HADOOP programing procedure, data combing has three stages: Map, Combine and Reduce. In Map stage, file blocks are received; read each record per line and process each record to get user id, item id and scores; after treatment, user id used as key, item id and scores used as value; output <key,value> pair. In Combine stage, it's designed to relieve communication burden between nodes caused by plentiful data generated in Map stage; with a hyphenated word, join up data which have common key to have local results and pass to Reduce function; likewise, user id used as key, item id and scores used as value; output <key,value> pair.

In Reduce stage, Reduce is responsible for integrated processing of data which were processed by several Maps in Map stage to get all rating values of each user. Likewise, user id used as key, item id and rating value as value.

Following previous thought, we design TidyDriver class, TidyMapper class, TidyCombiner class and TidyReducer class to realize data combing.

5.1.1. Tidydriver Class

TidyDriver class is entry of data combing. Initialize Job in through TidyDriver class; set the actual implementation class of Map, Combine and Reduce operation through setMapperClass, setCombinerClass and setReducerClass function. Add input file catalog through addInputPath; set result output catalog through setOutputPath.

5.1.2. Tidymapper Class

This class is actual implementation of Map operation. After segmentation of big data by HADOOP platform, data blocks are passed to each sub node which executes Map operation. Map reads each record by row, in which user id, item id and rating value are processed through character string; then user id used as key; item id and rating value as value; output <key,value> pair.

5.1.3. Tidycombiner Class

TidyCombiner class aims to perform local integration of data generated in Map and prevent Map from producing too many small data to cause communication burden. Combine operation joins locally data which have same user id with hyphenated number; so only one record of each user is transited to Reduce operation.

5.1.4. TidyReducer Class

TidyReducer class receives data processed by other nodes and makes global merger, generating all rating records of each user. TidyReducer inputs such records to HDFS, making preparations for calculating similarity in the future.

5.2. Prediction of Rating

Rating prediction is for user unrated items by means of similar neighbors' evaluation values. The required nearest neighbors are found from the closest Job. During Job initialization of rating predication, the nearest neighbor is given to calculate the path of Job output result; at Map stage, calculate user u's rating of unrated item j; then at Reduce stage, output predicted rating value.

To address the request of huge data, rating prediction request can be passed to several sub nodes, in good response to user request. Based on analysis, ForecaseMapper class and ForecastReducer class are devised.

5.2.1. ForecastMapper Class

This class is designed to realize Map process. User id and item id are figured out as per read records. After user id is got, read user u's nearest neighbor set from the output path of the 2nd Job; meanwhile, it's necessary to read the nearest neighbor's rating of item j from the 1st Job. With setup function, we can determine whether user nearest neighbor set and user scoring set are loaded to the memory all at once. If yes, load them all to the memory at the initialization stage, avoiding extended time delay when the nearest neighbor is being acquired or the neighbor reads or writes file to item rating; if no, use Cache mechanism to enhance query efficiency. Every time when the nearest neighbor set is inquired or the neighbor is evaluating item, it's necessary to check firstly whether there's record in the memory; if no, read file to get the record and save it in the memory; if the record can't be saved in the memory, eliminate records which are recently put in the memory by first-in first-out mechanism.

5.2.2. ForecastReducer Class

ForecastReducer class is responsible for outputting predicted rating value to HDFS file. The system makes recommendation to user according to output rating.

6 Experiment Design and Discussion

6.1. Experimental Data

The experimental data set is chosen from MovieLens. MovieLens [9] was founded in 1997, a project planned by GroupLens Research laboratory, focusing on collaborative filtering technologies for recommending movie to users. With user rated movies as blue print, MovieLens predict user rating of unrated movies as to introduce movies to viewer. To enhance the accuracy of predication, when a new user joins MovieLens, it needs to evaluate 15 types of movies.

We chose from MovieLens the dataset for the experiment, where the file named u.data includes totally 20000067 rating records by 71567 audiences about 10687 movies; each of them commented more than 15 movies, rating range is [1-5]. The file u.item contains information like movie id, movie name, release time, url and film type. Here we focused only the film type. In MovieLens, films are classified into 19 types including Crime, Action,Adventure,Animation and Comedy. It is shown in Table4.

Table 4. U.data Data Format

Field	Significance	Data type	Range
user	User id	Int	[1,67899]
item	Movie id	Int	[1,20691]
Rating value	Rating	Int	[1,5]

Divide 100000 records of rating made by 943 users in the whole dataset into training set and testing set, where training set takes 80% of the whole, *i.e*. 80000 records; testing set takes 20%, *i.e*. 20000 records; moreover, there is no intersection between training set and testing set. In paper [10], the author mentioned that user ratings are extremely sparse on large-scale e-commerce websites. So here it introduced the conception of sparsity [11] to examine MovieLens dataset. It is shown in Formula6.

$$\text{Sparsity}=1-\frac{\text{Total rating}}{\text{Number of users}\times\text{Total item}}\times 100\% \tag{6}$$

The sparsity of the data set is $1-\frac{100000}{943\times 1682}\times 100\% = 93.7\%$

6.2. Evaluation Criteria

To evaluate the quality of recommendation system, MAE (Mean Absolute Error) and RMSE (Root Mean Square Error) are often adopted, by calculating errors between predicted and scores to measure the accuracy of recommendation results. Suppose predicted rating is $\{p_1, p_2, p_3, \ldots p_n\}$ by user i, and real rating is $\{r_1, r_2 . r_3, \ldots r_n\}$, then MAE is defined as:

$$MAE = \frac{\sum_{i=1}^{n} | p_i - r_i |}{n} \tag{7}$$

$$RSME = \sqrt{\sum_{i=1}^{n} (p_i - r)^2} \tag{8}$$

However, no matter which is, MAE or RSME can only measure the overall performance of the recommendation system, occurring the problem of Heavy User [12]. To validate more accurately the effectiveness of such system, we define accuracy of single article recommendation, user recommendation accuracy and whole accuracy [13].

6.3. Selection of Similarity Formula

In the above parts, we introduced three kinds of similarity calculation formulae: cosine similarity formula, Pearson similarity formulae and modified cosine similarity formula. The variances exist among them for the calculation of similarity. In the same condition here, we test them and make comparison. Experimental data derived from MovieLens. According to the two-eight principle, training set is 80% and testing set is 20%. The experimental results are shown in Figure2.

From Figure 2, we see in the same condition, cosine similarity formula realized better prediction accuracy than the other two methods, possibly because when user ratings are too sparse, common ratings are very fewer. It's less reliable if the similarity between users is estimated based on commonly rated items. Hence in following test, we choose cosine similarity formula to do further testing.

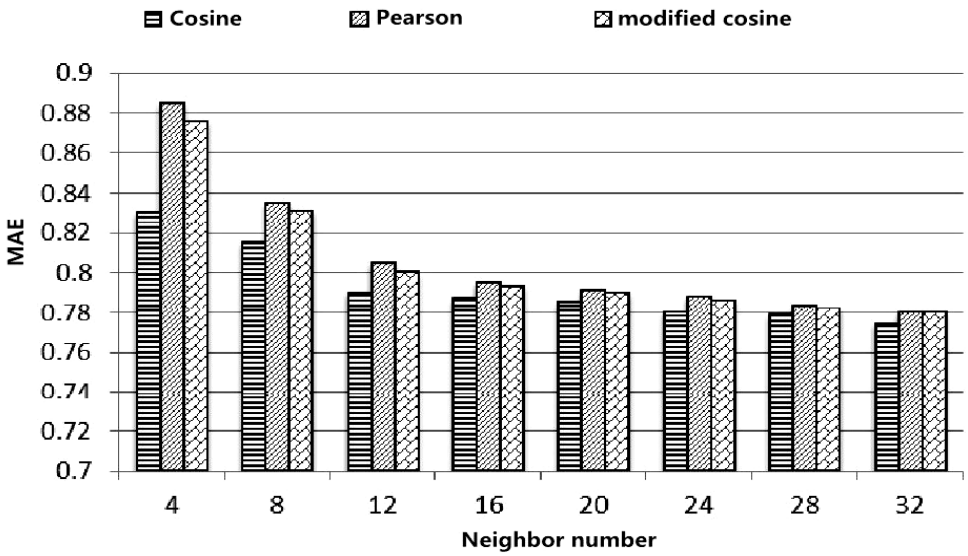


Figure 2. Experimental Comparison of the Three Similarity Formula

6.4. MAE Comparison of Algorithms

We compare prediction errors between the combination recommendation (Combination_CF) algorithm of recommendation algorithm based on attribute weight and weight prediction (BWApredict) and user similarity with traditional collaborative filtering algorithm. For Combination_CF algorithm, let threshold θ=1. Algorithm MAE contrast experimental results as shown in Figure3.

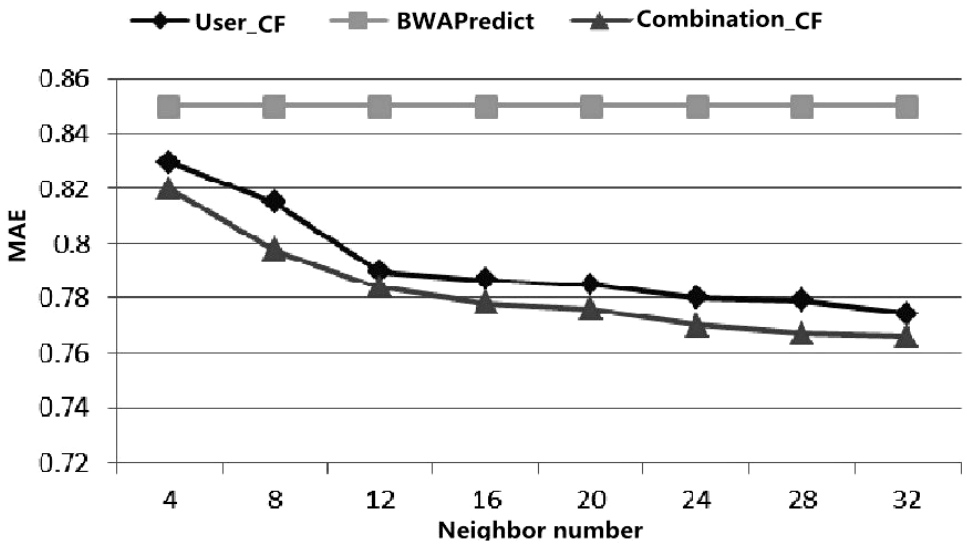


Figure 3. Three Algorithms MAE Contrast Experimental Results

From the experiment, we note the Combination_CF algorithm gained smaller prediction error than traditional collaborative filtering algorithm, which, however, is bigger than traditional method purely from the perspective of attribute weight. The method based on attribute weight can predict unrated records with the aid of rated records, which is more precise than primitive method, *i.e*. unrated record is zero. That is why the Combination_CF has high precision.

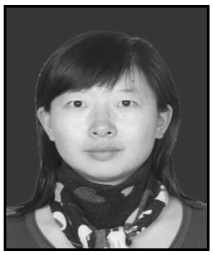
7. Conclusion

Starting with traditional collaborative filtering algorithm, the paper proposed two solutions to address the problem of collaborative filtering sparseness. One is rating prediction method based on attribute weight; the other is combination recommendation algorithm of attribute weight prediction and collaborative filtering. Experimental findings showed that the first method achieved more favorable prediction effect than that using default value because it considers user similarity; besides, the combination_CF algorithm performed better than traditional collaborative filtering. Considering the scalability of system, we tested traditional collaborative filtering algorithm on HADOOP platform, where it proved good speed-up ratio as experimental results suggested.

References

[1] X. Zhongyang, L. Qin, Z. Yufang and L. Wentian, "An improved collaborative filtering algorithm based on item classification", Study of computer application, vol. 2, **(2012)**, pp. 493-496.
[2] Z. Yang and S. Hua, "Based on neighbor users and neighboring projects to improve the collaborative filtering algorithm", Journal of Shenyang Normal University (Natural Science Edition), vol. 3, **(2012)**, pp. 382-385.
[3] X. Hong, P. Li, G. Aiyin and X. Yunjian, "Research on the improvement of collaborative filtering strategy based on the user's interest", Computer technology and development, vol. 4, **(2011)**, pp. 73-76.
[4] W. Hongchen, W. Xinjun, C. Yong and P. Zhaohui, "An improved recommendation algorithm based on collaborative filtering and partition clustering", Computer research and development, vol. 3, **(2011)**, pp. 205-212.
[5] Z. Yanlong, "An improved collaborative filtering algorithm based on social network analysis", South China University of Technology, **(2014)**.
[6] X. Yu, "Research on Collaborative Filtering Recommendation System of e-commerce personalized service", University of Electronic Science and technology, **(2013)**.
[7] P. Yu, C. Xiaoping and X. Yiping, "An improved collaborative filtering recommendation algorithm for Item-based", Journal of Southwestern University (Natural Science Edition), vol. 5, **(2007)**, pp. 146-149.
[8] X. Fuliang and J. Shuhao, "A content-based recommendation method based on collaborative filtering prediction and fuzzy similarity improvement", Modern library and information technology, vol. 2, **(2014)**, pp. 41-47.
[9] "Grouplens Research, Movielens: film recommendations", http://movielens.umn.edu
[10] B M Sarwar, "Sparsity, scalability, and distribution in recommender system", Minneapolis, MN: University of Minnesota, **(2001)**.
[11] G. Yanhong, "Collaborative filtering algorithm and application of recommendation system", Dalian: Dalian University of Technology, **(2008)**, pp. 13-55.
[12] P. Massa and P. Avesani, "Trust-aware Recommender System", Proceeding of the 2007 ACM conference on Recommender systems, **(2007)**.
[13] X. Chaolun, "Collaborative filtering recommendation algorithm based on social computing in electronic commerce", Zhejiang: Zhejiang University, **(2011)**, pp. 45-58.

Authors



Jingxia Guo, She received her M.S degree from Inner Mongolia Normal University. She is a lecturer in Bao Tou Medical College. Her research interests include Data mining.

Jinniu Bai, He received his B.S degree from Inner Mongolia Normal University and received his M.S degree from Nankai University. He is a professor in Bao Tou Medical College. His research interests include data mining.

