Hybrid Recommendation Algorithm Based on Latent Factor Model and PersonalRank

Jingjing Hu[1], Linzhu Liu[1], Changyou Zhang[2], Jialing He[1], Changzhen Hu[1]

[1] School of Software, Beijing Institute of Technology, China
[2] Institute of Software, Chinese Academy of Science, China
hujingjing@bit.edu.cn, 2220140568@bit.edu.cn, changyou@iscas.ac.cn, 1181710323@qq.com, czhoo@bit.edu.cn

Abstract

To promote the development of future social networks, an efficient recommendation algorithm is needed. Due to high-dimensional structures and missing rating information of the user-item matrix caused by sparse data, the direct use of a single recommendation algorithm is inefficient. In order to improve the accuracy of personalized recommendations, we propose an improved hybrid recommendation algorithm based on the latent factor model (LFM) and the PersonalRank algorithm, which adopts a cascade mixing approach with features expanding. First, it fills in a sparse matrix using LFM for the users who did not evaluate for the items. Next, it sets up a graph model and uses the PersonalRank algorithm in the filling matrix. Finally, it computes *PR* values of each user for items and implements sorting. The efficiency of the hybrid algorithm was verified on the MovieLens dataset. Compared with the PersonalRank algorithm, it can improve the accuracy rate and the recall rate of top-N recommendations.

Keywords: Latent factor model, PersonalRank, Hybrid recommendation algorithm

1 Introduction

PersonalRank originates from the PageRank algorithm, and PageRank is applied to the popular search engine Google [1-2]. It is a personalized recommendation algorithm based on graphing, and can transform the personalized recommendation problem into node selection in a graph. However, owing to data sparseness of the user-item rating matrix, use of this algorithm alone will predict ratings with low accuracy. Instead, it is necessary to use one of several effective strategies to fill in the sparse matrix [3-4].

Latent factor model (LFM) is a kind of algebraic model and calculation theory used for information retrieval and knowledge acquisition [5]. The model was proposed in the field of text mining earliest, which was used to analyze a large number of text sets obtained by statistical calculating and extract the latent semantic structure between words and words. The latent semantic structure representing words and texts can be used to remove the relevance between words and simplify text vectors so as to achieve the dimension reduction.

LFM usually creates a scoring matrix, or a user-item matrix. In this matrix, each row element corresponds to a user; each column element corresponds to an item; and each record indicates a user's rating of an item [6]. From the view of matrix decomposition, LFM decomposes the user's scoring matrix into two low dimensional matrices, which aims at approximately representing the scoring matrix. The two low dimensional matrices are learned by training the data and optimizing the loss function. The product of the two lower dimensional matrices can be used to predict the target user's ratings of the items. In recent years, LFM has obtained good application effect in top-N recommendations.

In an actually application system, it is hard to reach high precision rate for a single recommendation algorithm, so a variety of recommendation systems utilize mixed recommendation algorithms to improve the accuracy.

The main purpose of using hybrid recommendation algorithm is to alleviate the flaws of each recommendation algorithm itself. For example, the most traditional hybrid recommendation system takes use of a recommendation algorithm based on content recommending and collaborative filtering-based recommending [7]. The hybrid algorithm not only solves the problem of collaborative filtering algorithm about cold start, but also solves the problem of sparsity in the scoring matrix. And it also resolves the problem that content recommendation result is too single. It not only overcomes the limitation of each recommendation algorithm, but also, to a certain extent, improves the accuracy of the recommending. Therefore, it is the key point of this paper to implement an effective hybrid recommendation algorithm. Specifically, LFM and PersonalRank algorithm are mixed to obtain higher recommendation precision.

*Corresponding Author: Jingjing Hu; E-mail: hujingjing@bit.edu.cn
DOI: 10.3966/160792642018051903027

The structure of the paper is organized as follows. Section 1 illustrates the background and fundamental principles of recommendation algorithms. Section 2 describes the detailed algorithm of LFM and PersonalRank. Section 3 proposes a hybrid personalized recommendation algorithm based on LFM and PersonalRank. The efficiency of recommendation is verified in Section 4, and the final section presents the conclusions.

2 The Algorithm Basis

2.1 PersonalRank

PersonalRank is a graph-based recommendation algorithm that transforms the recommendation problem into node selection in graph [8-9]. Using a graph to present the rating behavior for users, the rating behavior can be seen as a series of tuples where each tuple (u,i) indicates the rating of user u for item i. Therefore, the data can be expressed by a bipartite graph.

G(V, E) represents a bipartite graph for user-item, where $V = V_U \bigcup V_I$ consists of the nodes for a set of users and items. For each tuple (u,i) in the data set, the graph has a set of edges $e(v_u, v_i)$, where $v_u \in V_U$ is the corresponding node of user u, and $v_i \in V_I$ is the corresponding node of item i. The bipartite graph is shown in Figure 1, in which the circle represents the user and the box indicates the item. It can be seen that there are records of past scoring for user *A* on items *a* and *b*, user *B* on items *b* and *d*, user *C* on item *a*, and user *D* on items *a*, *c*, and *d*.

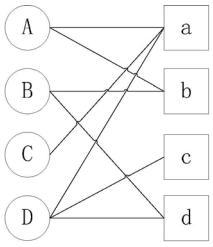

Figure 1. The user - item bipartite graph model

The basis of the PersonalRank algorithm is that the user u walks randomly from the starting node v_u. When wandering to a random node, the user first decides whether to continue walking or to terminate the walk and restart from the starting node v_u according to the probability $PR(i) = (1-\alpha)r_i + \alpha \sum_{j \in in(i)} \frac{PR(j)}{|out(i)|}$. The "*in(i)*" indicates the set pointing to *i* and the "*out(i)*" is the set that *i* is pointing to. If it needs to continue, a node will be selected to be the next node according to the uniform distribution randomly, and in this manner the user walks to the end. The probability of accessing each item node will converge to a number that is the final access probability for that item node [8]. This is shown in formula 1, where *PR* denotes the access probability of node *i*.

$$PR(i) = (1-\alpha)r_i + \alpha \sum_{j \in in(i)} \frac{PR(j)}{|out(i)|}, r_i = \begin{cases} 1, i = u \\ 0, i \neq u \end{cases} \quad \textbf{(1)}$$

From the formula (1), the main difference between PersonalRank and PageRank algorithm is that PersonalRank algorithm replaces *1/N* with r_i, where u represents the target user, which means that the probability from different points is different, so the calculated value is the correlation degree of user u relativity that user u relating to all the points.

The recommendation algorithm based on graph model can establish the relationship between users and objects better and more intuitively, and get the top-N recommendation list more naturally. However, because the algorithm requires continuous iteration on the binary graph to make the *PR* values of each node convergent in the graph. The time complexity of the algorithm is very high.

Therefore, PersonalRank is transformed into the form of matrix to take the place of the iterative method as shown in formula 2. M is the transition probability matrix of the binary graph, i.e.

$$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad \textbf{(2)}$$

In the matrix, $A \in R^{n \times m}$, n is the number of users, and m is the number of items. If user i has connection with items j, then the value of R_{ij} is assigned to the reciprocal of the degree of user node. In the matrix of $B \in R^{m \times n}$, if there is correlation between user i and item j, then the value of R_{ij} is assigned to the reciprocal of the degree of item node.

Then, the iterative formula 1 can be transformed into formula 3.

$$r = (1-\alpha)r_0 + \alpha M^T r \quad \textbf{(3)}$$

Where $r_0 = \begin{bmatrix} V \\ 0 \end{bmatrix}$, V is an n-dimensional vector, and for the user i, the values in the i-th row of the vector V is 1 and the values in other positions are 0.

The final solution is shown as in formula 4.

$$r = (1-\alpha)(1-\alpha M^T)^{-1} r_0 \quad \textbf{(4)}$$

So it only needs to calculate $(1-\alpha M^T)^{-1}$ once, but it needs to calculate the inverse matrix of sparse matrix rapidly.

2.2 Latent Factor Model

Latent factor model (LFM) builds on the relationship between users' interests and items to be

recommended by extracting latent features [10]. From the perspective of matrix factorization, two lower-dimensional matrices are divided by LFM. Learning the two matrices requires optimizing the cost function. Then, the two matrices can predict the scores for different items evaluated by target users.

Formula 5 is shown to calculate the interest of items for users.

$$R_{UI} = P_U Q_I = \sum_{k=1}^{K} P_{U,k} Q_{k,I} \quad \textbf{(5)}$$

The matrix R is the rating matrix, where value r_{ij} represents the score of $item_j$ to $user_i$. For a user, when the interest measure of all items is calculated, all items can be sorted to make additional recommendations. LFM algorithms extract several themes from a data set as a bridge to connect the users and items. Matrix R represents the product of the matrix P and Q.

P is a user-class matrix, where p_{ij} represents the interest measure of $class_j$ for $user_i$. Q is a class-item matrix, where q_{ij} represents the weight of $class_i$ for $item_j$.

To calculate the parameter values in the P matrix and Q matrix, the common method solves this parameter values by an optimization with cost function. The gradient descent algorithm is a kind of basic method for this optimization [11-12].

Learning the P and Q matrix can use the observed value of the training set directly by minimizing RMSE (Root Mean Square Error), where the cost function is defined as formula 6.

$$C(p,q) = \sum_{(u,i)\in Train} (r_{ui} - \overline{r_{ui}})^2 = \sum_{(u,i)\in Train} (r_{ui} - \sum_{f=1}^{F} p_{uf} q_{if}) \quad \textbf{(6)}$$

The direct optimization of the cost function may lead to over-fitting. Therefore, it is necessary to add the regular term $\lambda(\|p_u\|^2 + \|q_i\|^2)$, where λ is the regularization parameter, $\lambda \geq 0$. Thus the formula 7 for cost function is obtained.

$$C(p,q) = \sum_{(u,i)\in Train} (r_{ui} - \sum_{f=1}^{F} p_{uf} q_{if})^2 + \lambda(\|p_u\|^2 + \|q_i\|^2) \quad \textbf{(7)}$$

The above cost function in formula 7 defines two sets of parameters p_{uf} and q_{if}, and using the gradient descent method first requires computing the partial derivative of the cost function. Thus the formula 8 is obtained.

$$\begin{cases} \dfrac{\partial C}{\partial p_{uf}} = -2q_{ik} + 2\lambda p_{uk} \\ \dfrac{\partial C}{\partial q_{if}} = -2p_{uk} + 2\lambda q_{ik} \end{cases} \quad \textbf{(8)}$$

Then, according to the stochastic gradient descent algorithm, the parameters must move along the steepest descent direction, leading to formula 9.

$$\begin{cases} p_{uk} = p_{uk} + \beta(q_{ik} - \lambda p_{uk}) \\ q_{if} = q_{if} + \beta(p_{uk} - \lambda q_{ik}) \end{cases} \quad \textbf{(9)}$$

In the formula, β is the learning rate, obtained by repeated experiments. When the cost function converges to a stable value, the iteration ends and the P and Q matrix are generated. There are many methods to initialize the P and Q matrix, generally by filling the two matrices with a small range of random numbers. Many experiments showed that it is effective to have the random number be proportional to $1/sqrt(F)$, where F is the number of latent factors.

3 The Hybrid Algorithm based on LFM and PersonalRank

In general, the mixed modes include: cascade mixing, weighted mixing, switching mixing, feature expanding and so on [13].

Cascade mixing means that suppose there is a user-item predictor r, and for each user-item pair (u, i), a predicted value is given [14]. Then the next predictor can be designed on the basis of the former prediction. The output of the previous predictor can be the input of the latter predictor so as to increase the accuracy of the prediction.

The weighted mixing mode is that suppose there are n different predictors $\{r_1, r_2, r_3, ..., r_n\}$, and the simplest weighted method is to use linear weighting [15]. It is that the final predictor r is the linear weight of the n predictors, and the specific weight value of each predictor is obtained by repeated testing with test data so as to achieve the best recommendation results.

Switching Mixing: For different situations, the recommendation system may adopt different strategies [16]. Switching mixing can select a more suitable predictor by changing models according to the different circumstances.

Feature expanding is putting the information of characteristics generated by a recommendation algorithm into another algorithm to act as the input of the feature [17]. There are also methods for combinations of the features: assembling the features from different sources of recommendation, and applying them into another recommendation algorithm.

In the above-mentioned several methods, the cascade mixing mode is suitable for the predictor with depth structure, and weighted mixing as well as switching mixing is more suitable for the predictor with breadth structure. In this paper, we propose a recommendation algorithm that combines the expansion of latent factor feature and cascading with PersonalRank algorithm to improve the accuracy of recommendation.

This paper designs a hybrid algorithm based on LFM and PersonalRank, and adopts a cascade mixing approach with feature expanding, as shown in Figure 2.

First, it predicts users' rating for items which have not been scored by using LFM, filling in the ratings matrix with predictive ratings greater than a certain threshold. Next, it builds the graphical model of users and items, using the PersonalRank algorithm to compute the rank value of each item for each user. Finally, top-N is recommended.

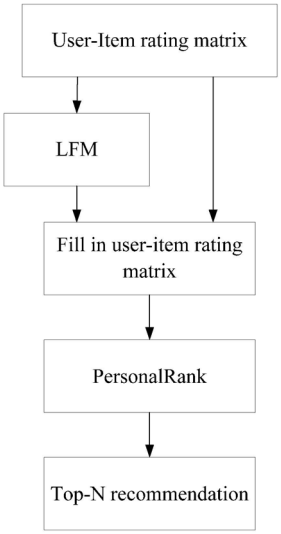

Figure 2. The structure of the hybrid algorithm based on LFM and PersonalRank

3.1 Sparse Matrix Filling

In the recommendation system, the number of users and commodities is very large, but the number of users' average browsing and scoring is relatively fewer. The sparsity of rating data cannot accurately analyze the user's preferences, and this can seriously influence the performance of the algorithm. Although it is essentially impossible to solve, there are strategies to alleviate the problem to some extent. One strategy is filling in values based on prediction, such as filling in a fixed default value or average value, or forecasting by BP neural network, naive Bayesian classification, or predicting based on the content. However, this method is time-consuming and calculation-intensive, and the filling value itself may generate error that will lower the recommended quality. Another approach is dimensionality reduction technology, including singular value decomposition (SVD) [18-19] and principal component analysis [20].

In this paper, LFM is used by predicting score to fill in the sparse matrix. To do this, LFM digs out the hidden factor of users and items fully by data set, rather than by explicit features. This allows mining of the characteristics of the unexplained, making the granularity of characteristics finer and the dimensions more broad so that it predicts the score more accurately.

The relationship between users and objects are divided into strong and weak associations. The strong correlation is that the user has a rating behavior. The weak association is that the predicted score by LFM is greater than a predefined threshold r. A sparse matrix is filled with weakly connected users and items. Weak correlation shows that the users may have more preference for this item.

3.2 Establishment of a Graphical Model

The relationship between a user and an item can be described as a structure of the graph model, $G= (V, E)$, where V represents a node set and E represents an edge set. Users and items are denoted by nodes and the connection between them is represented as a side. The weighted undirected graph is built, and the greater the weight of the edge is, the closer the shared relationship between the two nodes has. The PersonalRank algorithm based on random walk can calculate the relationship between the two nodes. The graphical model is established as shown in Figure 3. The left shows the graphical model for the data of user with items recorded by the score, and the right is modeled using LFM to add edges. The corresponding adjacency matrix is given in Figure 4. The weight between the user and the item node is set to the score or the prediction score.

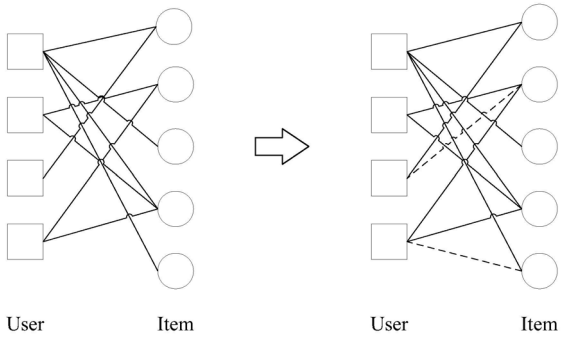

Figure 3. Using LFM to add edges

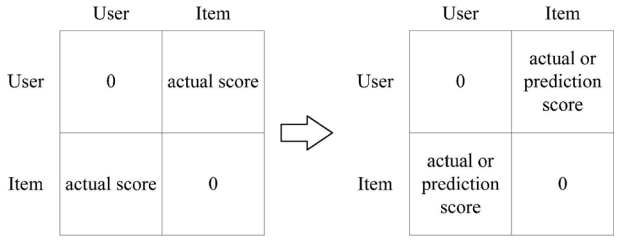

Figure 4. Adjacency matrix

3.3 Implementation of PersonalRank into the Graphical Model

In the hybrid algorithm, the iterative formula of the original PersonalRank is as shown in formula 10.

$$r^{(t+1)} = (1-\alpha)r_0 + \alpha S r^{(t)} \quad \textbf{(10)}$$

In the formula, S is the normalized adjacency matrix of the graph, which is derived from A. The edge weights of the user's rating for the item are defined and include the prediction score by LFM. A was defined as

the weight, as shown in formula 11.

$$A=(a_{ij})_{i,j\in V},\ a_{ij}=\begin{cases}\text{actual or prediction score } (i,j)\in E\\ 0 \qquad\qquad\qquad \text{otherwise}\end{cases} \quad (11)$$

The matrix A is symmetric and a_{ij} represents the relationship between node i and j. If there is $a_{ij}>a_{ij'}$, then the relationship between i and j is closer than the relationship between i and j'.

In the implementation, $r^{(0)}$ represents the initial distribution of nodes on the graph, all set to 1. If we want to recommend items for user i, node i will be set as the starting node. Here r_0 is a vector in which the element corresponding to the starting node is set to 1 and α is the probability. Through a number of iterations, each node will reach a stable value and then the nodes will be sorted by their stable value. The larger the value is, the more likely it is to match the target user.

4 Experiments and Performance Analysis

4.1 Data Sets and Evaluation Criteria

The MovieLens 10M data set [21] was used for testing. To ensure that each user had at least 20 movie scores, the 10 thousand samples included 84 users and 2976 movies, with sparseness of about 96%, and the100 or 500 thousand samples included 730 users and 6373 movies with a sparse degree of 97.85%. After the 10 thousand, 100 thousand and 500 thousand samples were extracted from this data set based on each user's score for an item in the order of time, the first 70% data acted as the training set and the latter 30% as the test set.

Since the top-N recommendation will give each user a recommended list, the precision rate and the recall rate are used to evaluate the accuracy of the recommendation. Here, $R(U, I)$ is the set of users U who were recommended with items I and $T(U, I)$ is the set of users u who have the actual score of the items. The accuracy of recommendation results is defined as shown in formula 12 and the recall rate is defined as formula 13.

$$Precision=\frac{\sum_{(u,i)\in Test}\left|R(u,i)\cap T(u,i)\right|}{\sum_{(u,i)\in Test}\left|R(u,i)\right|} \quad (12)$$

$$Recall=\frac{\sum_{(u,i)\in Test}\left|R(u,i)\cap T(u,i)\right|}{\sum_{(u,i)\in Test}\left|T(u,i)\right|} \quad (13)$$

4.2 The Experimental Results Comparison

Two algorithms for top-N recommendation were tested. The first experiment using PersonalRank(PR) for the 10 thousand, 100 thousand and 500 thousand data sets was performed as follows. First, forecast the scores for the films without scores for each user. Then perform top-N sorting. Finally, the first N recommended results were compared with the test set to obtain the precision rate and recall rate. In the experiment, β =0.8. The results with average values of 10 times experiments are shown in Table 1 to Table 3.

Table 1. 10 thousand samples by PR

	precision rate	recall rate
top5	0.121428	0.016787
top10	0.122619	0.033903
top20	0.101785	0.056287
top30	0.095634	0.079328

Table 2. 100 thousand samples by PR

	precision rate	recall rate
top5	0.336986	0.041437
top10	0.290410	0.071421
top20	0.240479	0.118283
top30	0.207031	0.152747

Table 3. 500 thousand samples by PR

	precision rate	recall rate
top5	0.539523	0.096463
top10	0.489295	0.153642
top20	0.446483	0.180564
top30	0.369804	0.246687

The second experiment used the hybrid algorithm based on LFM and PersonalRank (LFM+PR). For the three samples, the prediction scores greater than 4 were filled in the user - item rating matrix. In the experiment, α=0.02, λ=0.01, β =0.8. The results are average values of 10 times experiments. The top-N recommended results are shown in Table 4 to Table 6.

Table 4. 10 thousand samples by LFM+PR

	precision rate	recall rate
top5	0.132812	0.017087
top10	0.130543	0.034076
top20	0.110711	0.057612
top30	0.106001	0.080209

Table 5. 100 thousand samples by LFM+PR

	precision rate	recall rate
top5	0.359093	0.058449
top10	0.308410	0.087654
top20	0.259843	0.123039
top30	0.228742	0.165493

Table 6. 500 thousand samples by LFM+PR

	precision rate	recall rate
top5	0.639297	0.179622
top10	0.567234	0.190234
top20	0.527890	0.239783
top30	0.398353	0.280762

It can be seen from Figure 5 to Figure 10 that using LFM to get the score filling sparse matrix improved the accuracy of the prediction score. On average, the precision rate increased by 10% and the recall rate increased by 19%.

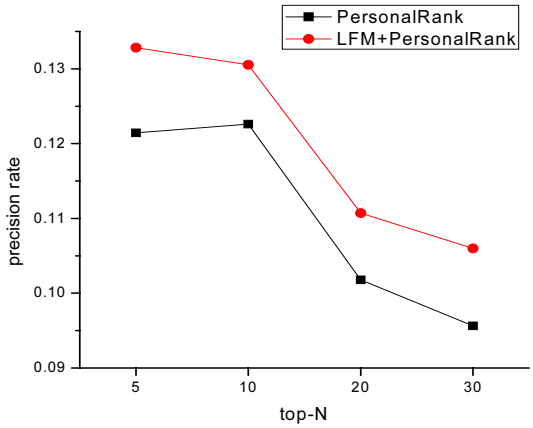

Figure 5. Comparison of the precision rate by LFM+PR and PR in 10 thousand samples

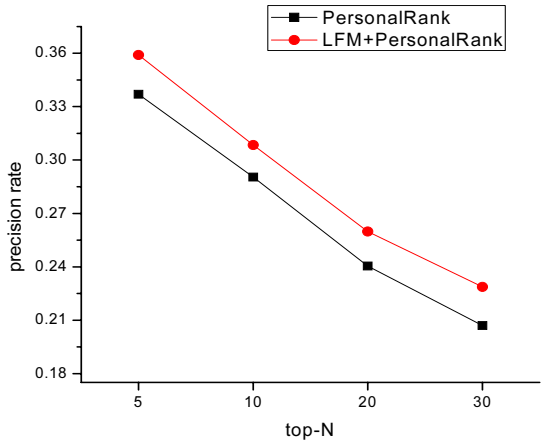

Figure 6. Comparison of the precision rate by LFM+PR and PR in 100 thousand samples

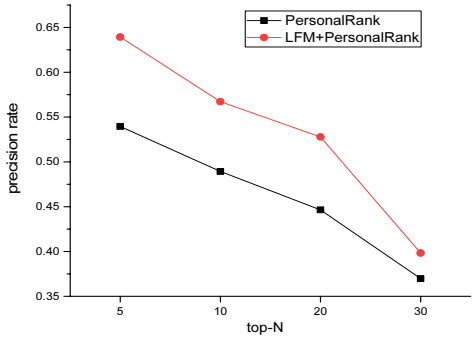

Figure 7. Comparison of the precision rate by LFM+PR and PR in 500 thousand samples

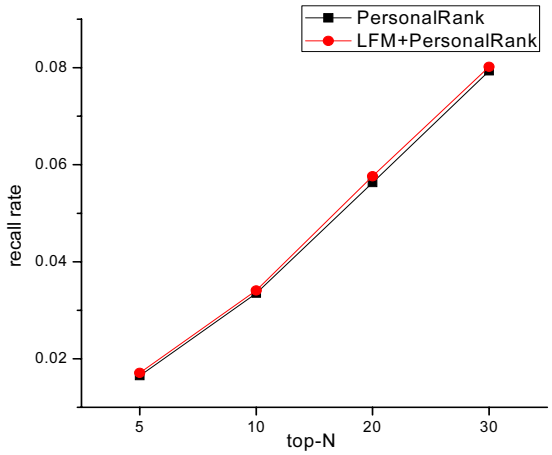

Figure 8. Comparison of the recall rate of LFM+PR and PR in 10 thousand samples

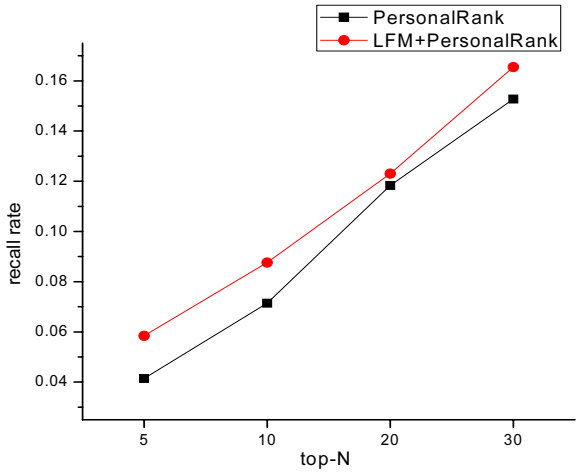

Figure 9. Comparison of the recall rate of LFM+PR and PR in 100 thousand samples

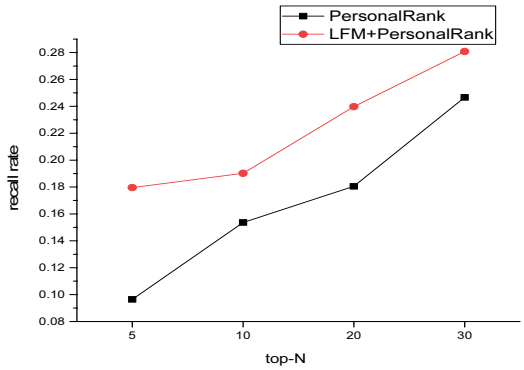

Figure 10. Comparison of the recall rate of LFM+PR and PR in 500 thousand samples

5 Conclusions

In this paper, a new hybrid recommendation algorithm is proposed that combines LFM and the

PersonalRank algorithm based on a bipartite graph. The algorithm can improve the accuracy of recommendations by first filling the sparse matrix before using the PersonalRank algorithm. Compared with other methods, this paper adopts the prediction score by LFM to fill the sparse matrix, which completely uses the data set to dig out the hidden factors of users and items, rather than using the explicit features. This approach helps discovering the unexplained features, which refines the granularity and broadens the dimensions of features. Experimental results show that filling the sparse matrix using LFM can effectively reduce the impact of data sparsity so as to predict the score more accurately in top-N recommendation.

Acknowledgements

This work has been supported by the National Key Research and Development Program of China (Grant No. 2016YFB0800700, 2017YFB1400902), the National Natural Science Foundation of China (Grant No. 61772070, 61672508, U1636213) and Beijing Higher Education Young Elite Teacher Project (Grant No. YETP1169).

References

[1] D.-V Daele, A. Kimmig, L.-D Raedt, PageRank, ProPPR, and Stochastic Logic Programs, *24th International Conference on Inductive Logic Programming*, Nancy, France, 2015, pp. 168-180.

[2] F. Tian, Y. Chen, X. Wang, T. Lan, Q. Zheng, K.-M. Chao, Common Features Based Volunteer and Voluntary Activity Recommendation Algorithm, *2015 IEEE 12th International Conference on E-Business Engineering (Icebe)*, Beijing, China, 2015, pp. 43-47.

[3] X. Zhao, Z. Niu, K. Wang, K. Niu, Z. Liu, Improving Top- *N* Recommendation Performance Using Missing Data, *Mathematical Problems in Engineering*, Vol. 2015, pp. 1-13, September, 2015.

[4] H. Jia, S. Ding, M. Du, Y. Xue, Approximate Normalized Cuts without Eigen-decomposition, *Information Sciences*, Vol. 374, pp. 135-150, December, 2016.

[5] D. Yu, Y. Liu, Y. Xu, Y. Yin, Personalized QoS Prediction for Web Services using Latent Factor Models, *2014 IEEE International Conference on Services Computing (Scc 2014)*, Anchorage, AK, 2014, pp. 107-114.

[6] C.-L. Liu, X.-W. Wu, Large-scale Recommender System with Compact Latent Factor Model, *Expert Systems with Applications*, Vol. 64, pp. 467-475, December, 2016.

[7] H.-H. Qiu, Y. Liu, Z.-J. Zhang, G.-X. Luo, An Improved Collaborative Filtering Recommendation Algorithm for Microblog Based on Community Detection, *Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Kitakyushu, Japan, 2014, pp. 876-879.

[8] T. Zhao, R. Xiao, C. Sun, H. Chen, Y. Li, H. Li, Personalized Recommendation Algorithm Integrating Roulette Walk and Combined Time Effect, *Journal of Computer Applications*, Vol. 34, No. 4, pp. 1114-1117, April, 2014.

[9] B. Shen, B.-W. Hu, H. Zhang, Method for the Analysis of the Preferences of Network Users, *IET Networks*, Vol. 5, No. 1, pp. 8-12, January, 2016.

[10] L. Qiu, S. Gao, W. Cheng, J. Guo, Aspect-based Latent Factor Model by Integrating Ratings and Reviews for Recommender System, *Knowledge-Based Systems*, Vol. 110, pp. 233-243, October, 2016.

[11] H. Lai, Y. Pan, Y. Tang, N. Liu, Efficient Gradient Descent Algorithm for Sparse Models with Application in Learning-to-rank, *Knowledge-Based Systems*, Vol. 49, pp. 190-198, September, 2013.

[12] L. Yan, Y.-J. Li, X. Yang, W. Gao, Gradient Descent Technology for Sparse Vector Learning in Ontology Algorithms, *Journal of Discrete Mathematical Sciences & Cryptography*, Vol. 19, No. 3, pp. 753-775, July, 2016.

[13] J. Chen, J. Li, J. Xiao, Y. Tang, H. Fu, A Hybrid Collaborative Filtering Model: RSVD Meets Weighted-Network Based Inference, *Journal of Internet Technology*, Vol. 17, No. 6, pp. 1221-1233, November, 2016.

[14] M. Salehi, I.-N. Kamalabadi, Hybrid Recommendation Approach for Learning Material based on Sequential Pattern of the Accessed Material and the Learner's Preference Tree, *Knowledge-Based Systems*, Vol. 48, pp. 57-69, August, 2013.

[15] X. She, Y. Cai, L. Dong, New Recommendation Algorithm based on Multi-objective Optimization, *Journal of Computer Applications*, Vol. 35, No. 1, pp. 162-166, January, 2015.

[16] V.-N. Zhao, M. Moh, T.-S. Moh, Contextual-aware Hybrid Recommender System for Mixed Cold-start Problems in Privacy Protection, *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity)*, New York, NY, 2016, pp. 400-405.

[17] M. Sattari, I.-H. Toroslu, P. Karagoz, P. Symeonidis, Y. Manolopoulos, Extended Feature Combination Model for Recommendations in Location-based Mobile Services, *Knowledge and Information Systems*, Vol. 44, No. 3, pp. 629-661, September, 2015.

[18] T.-M. Chang, W.-F. Hsiao, W.-L. Chang, An Ordinal Regression Model with SVD Hebbian Learning for Collaborative Recommendation, *Journal of Information Science & Engineering*, Vol. 30, No. 2, pp. 387-401, March, 2014.

[19] Q. Ba, X. Li, Z. Bai, Clustering Collaborative Filtering Recommendation System Based on SVD Algorithm, *Proceedings of 2013 IEEE 4th International Conference on Software Engineering and Service Science (Icsess)*, Beijing, China, 2012, pp. 963-967.

[20] N. Evangelopoulos, X. Zhang, V.-R. Prybutok, Latent Semantic Analysis: five methodological recommendations, *European Journal of Information Systems*, Vol. 21, No. 1, pp. 70-86, January, 2012.

[21] GroupLens, *MovieLens*, http://grouplens.org/datasets/movielens/.

Biographies

Jingjing Hu received the Ph.D. degree in computer science from Beijing Institute of Technology, China. She is currently a lecturer in the School of Software of Beijing Institute of Technology. Her research interests are in the areas of service computing, multi-agent systems, and parallel computing.

Linzhu Liu received his BEng degree from the School of Computer Science and Technology, Beijing Information Science and Technology University in 2014 and received his M.S. Degree in the School of Software, Beijing Institute ofTechnology in 2016. Her current research interests include machine learning, parallel computing and software engineering.

Changyou Zhang received his Ph.D degree in Beijing Institute of Technology, Beijing, China. He is a senior engineer and professor in Institute of Software, Chinese Academy of Science. His current research interests include intelligent information network, computer supported cooperative work, and parallel programming

Jialing He is a master student in the School of Software, Beijing Institute of Technology, China. She received her BEng degree from Beijing Institute of Technology in 2016. Her research interests include cloud security, data mining and software engineering.

Changzhen Hu received the Ph.D. degree from Beijing Institute of Technology. He is currently a professor in the School of Software, Beijing Institute of Technology. His current research interest is cyberspace security and web intelligence.