A topic community-based method for friend recommendation in large-scale online social networks

He, C, Li, H, Fei, X, Yang, A, Tang, Y & Zhu, J

Original citation & hyperlink:
He, C, Li, H, Fei, X, Yang, A, Tang, Y & Zhu, J 2016, 'A topic community-based method for friend recommendation in large-scale online social networks' *Concurrency and Computation: Practice and Experience*, vol 29, no. 6, e3924
https://dx.doi.org/10.1002/cpe.3924

DOI 10.1002/cpe.3924
ISSN 1532-0626
ESSN 1532-0634

Publisher: Wiley

CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2010; **00**:1–20
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

A topic community-based method for friend recommendation in large-scale online social networks

Chaobo He[1,4,*,†], Hanchao Li[2], Xiang Fei[2], Atiao Yang[3], Yong Tang[4] and Jia Zhu[4]

[1]*School of Information Science and Technology, ZhongKai University of Agriculture and Engineering, Guangzhou, 510225, China*
[2]*Department of Computing, Coventry University, Coventry, CV1 5FB, UK*
[3]*School of Mathematic and Computer Science, Guizhou Normal University, Guiyang, 550001, China*
[4]*School of Computer, South China Normal University, Guangzhou, 510631, China*

SUMMARY

Online social networks (OSNs) have become more and more popular and have attracted a great many users. Friend recommendation, which is one of the important service in OSN, can help users discover their interested friends and alleviate the problem of information overload. However, most of existing recommendation methods only consider either user link or content information and hence are not effective enough to provide high quality recommendations. In this paper, we propose a topic community-based method via Nonnegative Matrix Factorization (NMF). This method first applies joint NMF model to mine topic communities existing in OSN by combing link and content information. Then it computes user pairwise similarities and makes friends recommendation based on topic communities. Furthermore, this method can be implemented using the MapReduce distributed computing framework. Extensive experiments show that our proposed method not only has better recommendation performance than state-of-the-art methods, but also has good scalability to deal with the problem of friend recommendation in large-sale OSNs. Moreover, the application case demonstrates that it can significantly improve friend recommendation service in the real world OSN. Copyright © 2010 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: online social networks; topic community mining; friend recommendation; nonnegative matrix factorization; MapReduce

1. INTRODUCTION

Online social networks (OSNs), such as Facebook, LinkedIn and Twitter, have attracted billions of users and have become the most popular platforms to publish, share and obtain information. One of the important driving factors is that OSNs facilitate friendship establishment and maintenance among users. However, users are often difficult to discover their appropriate and relevant friends under the environment of information overload in large-scale OSNs and must rely heavily on friend recommendation service provided in OSNs (Figure 1). Recently, friend recommendation methods are mostly based on either link information (e.g., friendship or followership) [1, 2, 3] or content information (e.g., tag, post or profile) [4, 5, 6] existing in users. However, owing to using single information, these methods are not effective enough to recommend high quality friends to users. For example, Figure 2 shows an example of undirected OSN graph, where every user has his/her

*Correspondence to: Chaobo He, School of Information Science and Technology, ZhongKai University of Agriculture and Engineering, Guangzhou, 510225, China.
†E-mail: hechaobo@foxmail.com

own interest tag (e.g., OSN or Big data). If we only consider link information to make friend recommendation, user u_1 and u_6 will be recommended to each other to add as friends, because they have common friend u_5. Nevertheless, the possibility that they accept this recommendation may be slim for their different interest tags. On the other hand, if we only consider content information, u_2 and u_5 will not be recommended to each other for their different interest tags, but they have two common friends: u_1 and u_3 , and thus, they may be very interested in being friends. In real life, because of the principle known as homophily, people who share more social links and attributes have greater probability to become friends [7, 8]. Accordingly, to provide more accurate friend recommendation in OSNs we should both consider link and content information at the same time. Aiming to this requirement, we propose a kind of friend recommendation method based on topic community via joint Nonnegative Matrix Factorization (NMF). Users in the same topic community share more similarities among themselves, including link and content features, than those outside the community, and are more suitable to become friends. In particular, our method can be implemented using the MapReduce distributed computing framework and achieve good scalability in processing the large-scale OSN datasets. To the best of our knowledge, this is the first time to improve the performance of friend recommendation in OSN using the topic community-based method via joint NMF and our contributions are as follows:

- We design a topic community mining model that combines both link and content information using joint NMF. Using this model we develop a multiplicative iterative update algorithm to mine topic communities. Members in each topic community can be well guaranteed to share more similar link and content features. Based on the results of topic community mining we devise an algorithm to recommend Top-K friends to the target user. Moreover, for improving the scalability of our proposed method we implement the key algorithms based on the MapReduce distributed computing framework.
- Extensive experiments conducted on three real world OSN datasets demonstrate that the performance of our proposed method outperforms the state-of-the-art link-based and content-based friend recommendation methods. Furthermore, our method presents good scalability by running on the Hadoop cluster platform. The application case also shows that our method can improve friend recommendation service in OSNs very effectively.
- Our proposed method can address three classic problems existing in friend recommendation in large-scale OSNs effectively: cold start, data sparsity and scalability. It can make friend recommendation to cold start users based on their topic communities, reduce data sparsity by mining topic communities and increase scalability by running as MapReduce programs.

The rest of the paper is organized as follows. Section 2 discusses related work including NMF-based community mining and some representative methods of friend recommendation in OSNs. Section 3 describes the framework and key algorithms of our proposed method in detail. Section 4 presents the results of our experimental and application study in real world OSNs. Section 5 gives the conclusions and future work.

2. RELATED WORK

2.1. NMF-based community mining

NMF proposed by Lee and Seung is a low-rank matrix approximation model that focuses on the analysis of nonnegative data matrices [9]. Mathematically, NMF can be described as follows: given a matrix $X \in \mathbb{R}_{+}^{m \times n}$, then X can be decomposed into basis matrix W and coefficient matrix H, such that $X \approx WH^T$, where $W \in \mathbb{R}_{+}^{m \times r}$ and $H \in \mathbb{R}_{+}^{n \times r}$. r is preassigned and should satisfy $r \ll \min(m, n)$. W and H can be obtained by minimizing an objective function: $J(X, WH^T)$, where J is the objective function that measures dissimilarity between X and WH^T. The most widely used dissimilarity measure function is the Frobenius norm: $J(X, WH^T) = \| X - WH^T \|_F^2$.

NMF has been proven to be closely related to the K-means and spectral clustering methods and is widely used in the field of image analysis, text clustering and collaborative filtering

DOI: 10.1002/cpe

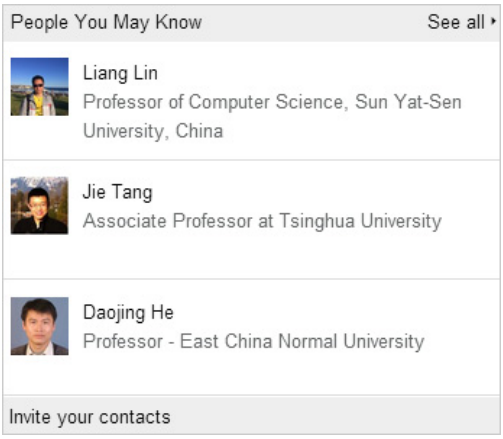

Figure 1. Friend recommendation service in LinkedIn OSN.

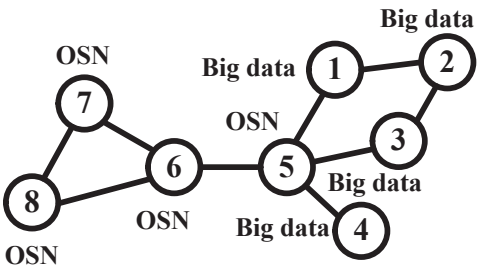

Figure 2. Toy example of an OSN.

[10, 11, 12, 13, 14]. Meanwhile, NMF-based method for community mining, which is a long-standing yet very difficult task in social network analysis [15], has also received a lot of attention. In [16], targeting undirected, directed and compound networks, Wang *et al.* proposed symmetric NMF, asymmetric NMF and joint NMF algorithms to mine community, respectively. The correctness and convergence properties of those algorithms were also studied. In [17], a Bayesian NMF model was presented to extract overlapping community from a network. This scheme had the advantage of soft-partitioning solutions and excellent module identification capabilities. Zhang *et al.* [18] proposed a so called bounded nonnegative matrix tri-factorization (BNMTF) method that can explicitly model and learn the community membership of each node as well as the interaction among communities. In [19], Lin *et al.* proposed a Meta Graph NMF framework named MetaFac that can extract community structures from various social contexts and interactions. In particular, MetaFac was also an online method to handle time-varying relations through incremental factorization. In [20], the authors explored how to apply the dictionary learning based on NMF to find the community structures in social networks. Experiment results conducted in both synthetic and real world datasets shown that this method was highly effective. The work in [21] developed a symmetric binary matrix factorization model (SBNMF) to identify overlapping communities. This model allowed us not only to assign community memberships explicitly to nodes, but also to distinguish outliers from overlapping nodes. In general, existing NMF-based methods for community mining mostly consider link information only, therefore, they are different from our method that considers both link and content information to mine topic communities.

2.2. Friend recommendation in OSN

There have been many methods presented for friend recommendation in OSN. Generally, these methods can be divided into three categories: link-based methods, content-based methods and hybrid methods. Link-based methods are based on the topological characteristics of OSN. For example, Zhu *et al.* [22] presented a link prediction-based approach to friend recommendation system by combining network topology and probabilistic relational model (PRM) approaches. The work in [23] proposed an algorithm for recommending relevant users that explores the topology of the network. This algorithm considered different factors that allowed us to identify candidate friends that can be considered good information sources. In [24], Carullo *et al.* also presented an effective friend recommendation algorithm that exploits the already existing links/relationships and the scalability is its major advantage. Content-based methods utilize user-generated content and recommend friends by pairwise user similarities. For example, Akcora *et al.* [25] proposed a method based on profile similarity to recommend similar friends and the similarity can be computed with different measures. Wan *et al.* [26] recommended friends according to the informational utility. It

DOI: 10.1002/cpe

viewed a post in social media as an item and utilized collaborative filtering techniques to predict the rating for each post. The candidate friends are then ranked according to their informational utility for recommendation. Hybrid methods can integrate link and content information together to make friend recommendation. For example, Wang *et al.* [27] presented a novel semantic-based friend recommendation system named Friendbook for social networks, which recommended friends to users based on all kinds of their life styles data including social links and life documents. By creating heterogeneous networks using user social links and attributes, Dong *et al.* [28] proposed a ranking factor graph model (RFG) for predicting friend links in social networks, which effectively improved the predictive performance. Our proposed method belongs to the hybrid one. However, our method recommends friends based on topic community discovered by integrating user link and content information. That is different from methods above and is actually more appropriate to reflect user preferences on friend selection.

3. PROPOSED FRAMEWORK

Our proposed framework comprises three main phases. The first phase constructs user link matrix and content feature matrix after data extraction and preprocessing. The second phase mines topic communities via joint NMF model so that user membership can be determined according to his strength distribution over given communities. The third phase computes pairwise user similarities in each community to generate a list of candidate friends. We then combine these candidate lists to obtain the Top-K recommendation friends for each target user. The overall framework of our proposed method is shown in Figure 3. Before we describe the details of each phase, in Table I we summarize the major notations used in the following sections.

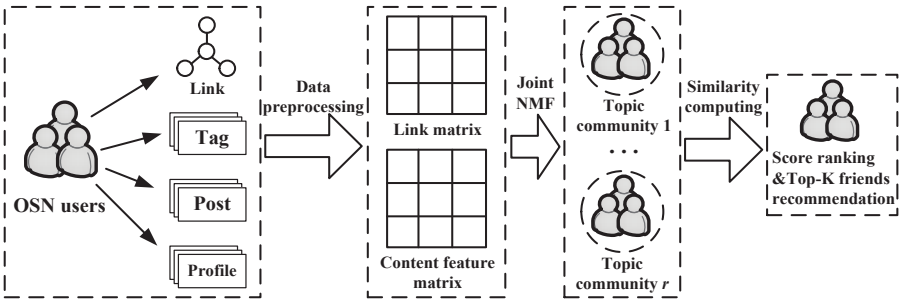

Figure 3. Topic community-based friend recommendation framework.

Table I. Notations used.

Notation	Description
$U = \{u_1, u_2, ..., u_n\}$	The set of users
e_{ij}	A linkage from u_i to u_j
$E = \{e_{ij} \| u_i \in U \wedge u_j \in U\}$	The set of all linkages between users
$A = \{a_1, a_2, ..., a_m\}$	The set of content features
$X = [x_{ij}]^{n \times n} \in \mathbb{R}_+^{n \times n}$	The user link matrix
$Y = [y_{ij}]^{m \times n} \in \mathbb{R}_+^{m \times n}$	The content feature-user matrix
r	The number of topic community
$C = \{c_1, c_2, ..., c_r\}$	The set of topic community
λ	The coefficient of the regularization

3.1. Topic community mining using joint NMF

Formally, OSN which comprises link and content information can be denoted as $OSN = \{U, E, A, X, Y\}$. Without loss of generality, we model the corresponding OSN link graph as a

DOI: 10.1002/cpe

directed unweighted graph. Namely, for $\forall x_{ij} \in X$, if $e_{ij} \in E$, then $x_{ij} = 1$, else $x_{ij} = 0$. Because content information, which includes user tag, post and profile, belongs to text type, the bag-of-words model can be applied to extract content features. After filtering stop words, we use the tf-idf (term frequency-inverse document frequency) [29] of each word as a content feature value for corresponding users. Namely, for $\forall y_{ij} \in Y$, we compute tf-idf of feature a_i in u_j content information text as its value. After data preprocessing above, we can obtain two nonnegative matrices: X and Y, which are both fit for factorization using NMF separately. To employ both the link and content information for mining topic community we combine X and Y into the following joint NMF model:

$$\min J(H,S,W) = \frac{1}{2}(\alpha||X - HSH^T||_F^2 + (1-\alpha)||Y - WH^T||_F^2 + \lambda(||H||_F^2 + ||S||_F^2 + ||W||_F^2)) \tag{1}$$

where $H \in \mathbb{R}_+^{n\times r}$, $S \in \mathbb{R}_+^{r\times r}$ and $W \in \mathbb{R}_+^{m\times r}$ are the topic community indicator, the topic community internal-strength indicator and the topic word affiliation indicator matrices, respectively. $\alpha \in [0,1]$ is a hyper-parameter that controls the importance of each factorization. Using the property of Frobenius norm and the matrix trace, the objective function J can be rewritten as follows:

$$\begin{aligned} J = \frac{1}{2}(&\alpha tr(XX^T) - \alpha tr(XHS^TH^T) - \alpha tr(HSH^TX^T) + \alpha tr(HSH^THS^TH^T) \\ &+(1-\alpha)tr(YY^T) - 2(1-\alpha)tr(YHW^T) + (1-\alpha)tr(WH^THW^T) \\ &+\lambda tr(H^TH) + \lambda tr(S^TS) + \lambda tr(W^TW)) \end{aligned} \tag{2}$$

Let $H = [h_{ij}]^{n\times r}$, $S = [s_{pq}]^{r\times r}$ and $W = [w_{ab}]^{m\times r}$, then the minimization problem of J can be restated as follows: minimizing J with respect to H, S and W under the constraints of $h_{ij} \geq 0$, $s_{pq} \geq 0$ and $w_{ab} \geq 0$. This is a typical constraint optimization problem and can be solved using the Lagrange multiplier method. Let δ_{ij}, φ_{pq} and θ_{ab} be the Lagrange multipliers for constraint $h_{ij} \geq 0$, $s_{pq} \geq 0$ and $w_{ab} \geq 0$, respectively, then the Lagrange function L is:

$$L = J + \delta tr(H^T) + \varphi tr(S^T) + \theta tr(W^T) \tag{3}$$

where $\delta = [\delta_{ij}]^{n\times r}$, $\varphi = [\varphi_{pq}]^{r\times r}$ and $\theta = [\theta_{ab}]^{m\times r}$. The derivatives of L with respect to H, S and W are:

$$\begin{aligned} \frac{\partial L}{\partial H} = &\alpha(HSH^THS^T + HS^TH^THS - XHS^T - X^THS) \\ &-(1-\alpha)(Y^TW - HW^TW) + \lambda H + \delta \end{aligned} \tag{4}$$

$$\frac{\partial L}{\partial S} = \alpha(H^THSH^TH - H^TXH) + \lambda S + \varphi \tag{5}$$

$$\frac{\partial L}{\partial W} = (1-\alpha)(WH^TH - YH) + \lambda W + \theta \tag{6}$$

Let $\partial L/\partial H = 0, \partial L/\partial S = 0$ and $\partial L/\partial W = 0$, and follow the KKT complementary slackness condition $h_{ij}\delta_{ij} = 0, s_{pq}\varphi_{pq} = 0$ and $w_{ab}\theta_{ab} = 0$, we derive the following multiplicative update rules for h_{ij}, s_{pq} and w_{ab}:

$$h_{ij} = h_{ij}\frac{[\alpha(XHS^T + X^THS) + (1-\alpha)Y^TW]_{ij}}{[\alpha(HSH^THS^T + HS^TH^THS) + (1-\alpha)HW^TW + \lambda H]_{ij}} \tag{7}$$

$$s_{pq} = s_{pq}\frac{[\alpha H^TXH]_{pq}}{[\alpha H^THSH^TH + \lambda S]_{pq}} \tag{8}$$

$$w_{ab} = w_{ab}\frac{[(1-\alpha)YH]_{ab}}{[(1-\alpha)WH^TH + \lambda W]_{ab}} \tag{9}$$

It has been proven by Lee and Seung that the objective function J is non-increasing under the above iterative updating rules, and that the convergence of the iteration is guaranteed [9]. When the

DOI: 10.1002/cpe

objective function J converges, we can get the local optimal solution for H, S and W, respectively. The community membership of every user can be identified based on H. Namely, $\forall h_{ij} \in H$ represents the strength of u_i belongs to community c_j and we can obtain the result of community mining by assigning every user to his maximum membership strength community. Moreover, the topic of every community can be described using representative topic words extracted from W. The topic community mining algorithm is summarized in Algorithm 1.

Algorithm 1 Topic community mining using joint NMF

Input: $X, Y, U, r, \alpha, \lambda$;
Output: $C = \{c_1, c_2, ..., c_r\}$;
1: Initialize H, S and W with random nonnegative values;
2: **repeat**
3: Update H, S and W based on equation 7, 8 and 9, respectively;
4: **until** stopping condition;
5: $\forall i = 1...r\ c_i \leftarrow \emptyset$;
6: **for each** $u_i \in U$ **do**
7: $p = \underset{l}{\operatorname{argmax}}\, h_{il}$;
8: $c_p = c_p \bigcup \{u_i\}$;
9: **end for**
10: **for each** $c_i \in C$ **do**
11: Extract representative topic words based on corresponding column vector $W(:, i)$;

3.2. Friend recommendation based on topic community

After discovering topic communities, the next phase is to generate candidate friends from these communities for recommendation. Users in the same topic community share more similar link and content features, so they are better representatives with similar interests in comparison with those outside the community and could be preferentially selected as candidate friends to the target user. The rank score of every candidate friend can be computed using a community-based similarity measure. Namely, given the candidate friend u_i and the target user u_j, the rank score $score(i, j)$ can be computed as follows:

$$score(i, j) = [HH^T]_{ij} \tag{10}$$

where H is derived by Algorithm 1. Finally, we sort these scores and output the Top-K friends to recommend to u_j. The complete recommendation procedure is summarized in Algorithm 2.

Algorithm 2 Friend recommendation based on topic community

Input: X, H, C, U;
Output: Ranked recommendation list;
1: Compute community-based similarity matrix $Q = HH^T$;
2: **for each** $u_i \in U$ **do**
3: Obtain the community c which u_i belongs to;
4: **for each** $u_j \in c \wedge x_{ij} = 0$ **do**
5: $score(i, j) \leftarrow [Q]_{ij}$;
6: $Ranklist.add(i, j, score(i, j))$;
7: **end for**
8: **end for**
9: Return the ranked lists of friends for each user;

We use the toy example of an OSN in Figure 2 to illustrate our method. Let $U = \{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8\}$, $A = \{OSN, BigData\}$, $r = 2$, $\alpha = 0.5$ and $\lambda = 1$, using joint NMF model discussed in section 3.1 we can obtain matrices H, S and W after convergence:

DOI: 10.1002/cpe

$$H^T = \begin{bmatrix} 0.04 & 0.00 & 0.04 & 0.08 & 0.45 & 0.87 & 0.76 & 0.76 \\ 0.68 & 0.69 & 0.68 & 0.41 & 0.64 & 0.11 & 0.00 & 0.00 \end{bmatrix}, S = \begin{bmatrix} 0.81 & 0.03 \\ 0.03 & 0.82 \end{bmatrix}, W = \begin{bmatrix} 0.11 & 0.89 \\ 0.82 & 0.00 \end{bmatrix}.$$

Matrix H clearly indicates that the topic community $c_1 = \{u_1, u_2, u_3, u_4, u_5\}$ and $c_2 = \{u_6, u_7, u_8\}$. H also advises that u_5 should be an overlapping user due to its significant contribution to both communities. Matrix S indicates that each detected topic community attains its perfect internal strength. Matrix W indicates that the strongly correlated topic word of c_1 is "Big Data" and c_2 is "OSN". In the recommendation phase, we compute the community-based similarity matrix $Q = HH^T$ and then can produce candidate friends for any user using Algorithm 2. For example, the ranked friend recommendation list of u_5 includes u_2 and his corresponding rank score is 0.45. Although u_2 has different interest tag, but its rank score is the highest by combining link features. For u_1, his ranked friend recommendation list is $\{u_3, u_4\}$ and the corresponding rank score is 0.47 and 0.29, respectively. Although u_6 has common friend u_5 with u_1, he is not recommended to u_1 by combining content features. Actually, u_6 and u_1 are not in the same community and the rank score of u_6 to u_1 is 0.11, which is much smaller than those of u_4 and u_3. Based on the illustrative example above, we can see that our proposed method is feasible to make friend recommendation by combining link and content information via joint NMF. It is also effective to address cold start and data sparsity problems that often persecute friend recommendation in OSNs. Although cold start users have no link information, by using content information extracted from their profiles they are still be grouped into topic communities to find similar users. Extensive researches on information recommendation have indicated that reducing data sparsity is a good strategy to improve recommendation performance [30, 31]. Our method also has this feature by using low-rank matrix factorization. For the toy example above, the sparsity of X, Y and H is 0.72, 0.44 and 0.19, respectively. It is obviously that the sparsity of H is much lower than that of X and Y.

3.3. Improving scalability using MapReduce

It is not hard to recognize that updating H, S and W iteratively is the most time-consuming task in Algorithm 1 and Algorithm 2. If we use the simple matrix multiplication, the approximate time complexity of updating H, S and W in each iteration is $O(n^2r + nr^2 + rm^2 + nmr + nr)$, $O(n^2r + r^3 + r^2)$ and $O((m + r)n^2 + mr^2 + mr)$, respectively. Although we can use sparse matrix multiplication or other optimization strategies to further decrease the time complexity, the computational cost is still heavy in large-scale OSNs. Therefore, improving the scalability of updating H, S and W iteratively is very necessary. Some studies have proved that the MapReduce distributed computing framework is fit for dealing with the problem of large-scale matrix multiplication [32, 33, 34], so multiplicative update rules for H, S and W can also be implemented using MapReduce. Considering the computation principles of multiplicative update rules for H, S and W are the same, due to space limitations we only present the implementation schema of multiplicative update rule for H (namely Equation 7) using MapReduce.

The MapReduce programming model was designed to simplify the processing of large files on a parallel system through user-defined Map and Reduce operations. A MapReduce job consists of two phases: a Map phase and a Reduce phase. During the Map phase, the Map operation transforms the input data into $< key, value >$ pairs in parallel. These pairs can be stored and shuffled by the system so as to accumulate all values for each key. During the Reduce phase, the Reduce operation is used to perform aggregations of all values associated with the same intermediate key in parallel and finally output the results in the form of $< key, value >$ pairs [33]. Obviously, data processed using the MapReduce must be presented as key-value pairs. In Equation 7, matrices X and Y are all sparse and they can be naturally stored in the format of $< i, j, x_{ij} >$ and $< i, j, y_{ij} >$ key-value pairs, respectively. For maximizing the data locality and minimizing the communication cost in computing Equation 7, we partition dense matrices H and W along the short dimension and this

DOI: 10.1002/cpe

partition renders the following view of H and W.

$$H = \begin{pmatrix} h_1 \\ h_2 \\ \cdots \\ h_n \end{pmatrix} \quad and \quad W = \begin{pmatrix} w_1 \\ w_2 \\ \cdots \\ w_m \end{pmatrix} \tag{11}$$

where $h_i (1 \leq i \leq n)$ and $w_j (1 \leq j \leq m)$ are all r-dimension row vectors. Consequently, H and W can be stored in the format of $< i, h_i >$ and $< j, w_j >$ key-value pairs, respectively. In particular, S is a $r \times r$ small matrix and hence can be directly stored in the distributed cache of machine clusters. For maximizing parallelism, we divide Equation 7 into 6 components: $P_1 = \alpha(XHS^T + X^THS)$, $P_2 = (1-\alpha)Y^TW$, $P_3 = \alpha(HSH^THS^T + HS^TH^THS)$, $P_4 = (1-\alpha)HW^TW$, $P_5 = \lambda H$ and $H = H_{\cdot} * (P_1 + P_2)_{\cdot}/(P_3 + P_4 + P_5)$, where every component can be computed using MapReduce. The entire flowchart of updating H on MapReduce cluster is depicted in Figure 4 and the computing process of every component is described as follows, respectively.

(1) Computing $P_1 = \alpha(XHS^T + X^THS)$. We first compute XH and X^TH. Let $T_1 = XH$ and $T_2 = X^TH$. Further let T_{1i} and T_{2i} denote the ith row vector of T_1 and T_2, respectively, then $T_{1i} = \sum_{j=1}^{n} x_{ij}h_j$ and $T_{2i} = \sum_{j=1}^{n} x_{ji}h_j$, which can be implemented by two sets of MapReduce operations at the same time.

- Map-I: map $< j, (i, x_{ij}) >$ and $< j, h_j >$ on j such that tuples with the same j are shuffled to the same machine in the form of $< j, \{h_j, (i, x_{ij})\} >$.
- Reduce-I: read $< j, \{h_j, (i, x_{ij})\} >$, emit $< (T_1, i), x_{ij}h_j >$ and $< (T_2, i), x_{ji}h_j >$.
- Map-II: map $< (T_1, i), x_{ij}h_j) >$ and $< (T_2, i), x_{ji}h_j) >$ such that tuples with the same (T_1, i) or (T_2, i) are shuffled to the same machine in the form of $< i, T_{1i} >$ or $< i, T_{2i} >$.
- Reduce-II: read $< i, \{T_{1i}, T_{2i}\} >$, emit $< i, P_{1i} >$, where $P_{1i} = \alpha(T_{1i}S^T + T_{2i}S)$.

(2) Computing $P_2 = (1-\alpha)Y^TW$. Let P_{2i} denote the ith row vector of P_2, then $P_{2i} = \sum_{j=1}^{m}(1-\alpha)y_{ji}w_j$. P_{2i} can also be implemented by two sets of MapReduce operations: Map-III, Reduce-III, Map-IV and Reduce-IV, which are similar to those used in computing X^TH above and the operation Reduce-IV finally emits $< i, P_{2i} >$.

(3) Computing $P_3 = \alpha(HSH^THS^T + HS^TH^THS)$. Let $T_3 = HS, T_4 = H^THS^T, T_5 = HS^T$ and $T_6 = H^THS$. Because T_4 and T_6 are all $r \times r$ small matrix, it is better to compute T_4 and T_6 first to greatly reduce intermediate data. T_4 and T_6 can be implemented by single MapReduce operation at the same time.

- Map-V: map $< i, h_i >$ to $< 0, h_i^Th_i >$, where 0 is a dummy key value for data shuffling.
- Reduce-V: read $< 0, \{h_i^Th_i\} >$, compute $T' = \sum_{i=1}^{n} h_i^Th_i$, emit $< T_4, (i, T_i'S^T) >$ and $< T_6, (i, T_i'S) >$ where T_i' is the ith row vector of T'. Next, we can copy T_4 and T_6 to all the machines that host h_i and compute P_3 using single MapReduce operation.
- Map-VI: read $< i, h_i >$, emit $< i, h_iST_4 >$ and $< i, h_iS^TT_6 >$.
- Reduce-VI: read $< i, \{h_iST_4, h_iS^TT_6\} >$, emit $< i, P_{3i} >$, where $P_{3i} = \alpha(h_iST_4 + h_iS^TT_6)$.

(4) Computing $P_4 = (1-\alpha)HW^TW$. Similar to the computing of P_3, we first compute $T_7 = W^TW$ and then compute $P_4 = (1-\alpha)HT_7$. The computing T_7 and P_4 all need single MapReduce operation: Map-VII, Reduce-VII, Map-VIII and Reduce-VIII, which are similar to those used in computing HSH^THS^T above and the operation Reduce-VIII finally emits $< i, P_{4i} >$.

(5) Computing $P_5 = \lambda H$. It only needs single MapReduce operation to compute P_5.

- Map-IX: read $< i, h_i >$,
- Reduce-IX: emit $< i, P_{5i} >$, where $P_{5i} = \lambda h_i$.

(6) Computing $H = H_{\cdot} * (P_1 + P_2)_{\cdot}/(P_3 + P_4 + P_5)$. Updating H can be parallelized through the single MapReduce operation.

- Map-X: read $< i, h_i >$, $< i, P_{1i} >$, $< i, P_{2i} >$, $< i, P_{3i} >$, $< i, P_{4i} >$ and $< i, P_{5i} >$ on i such that tuples with the same i are shuffled to the same machine in the form of $< i, \{h_i, P_{1i}, P_{2i}, P_{3i}, P_{4i}, P_{5i}\} >$.

DOI: 10.1002/cpe

- Reduce-X: read $< i, \{h_i, P_{1i}, P_{2i}, P_{3i}, P_{4i}, P_{5i}\} >$ and emit $< i, h_i^{new} >$, where $h_i^{new} = h_{i.} * (P_{1i} + P_{2i})./(P_{3i} + P_{4i} + P_{5i})$.

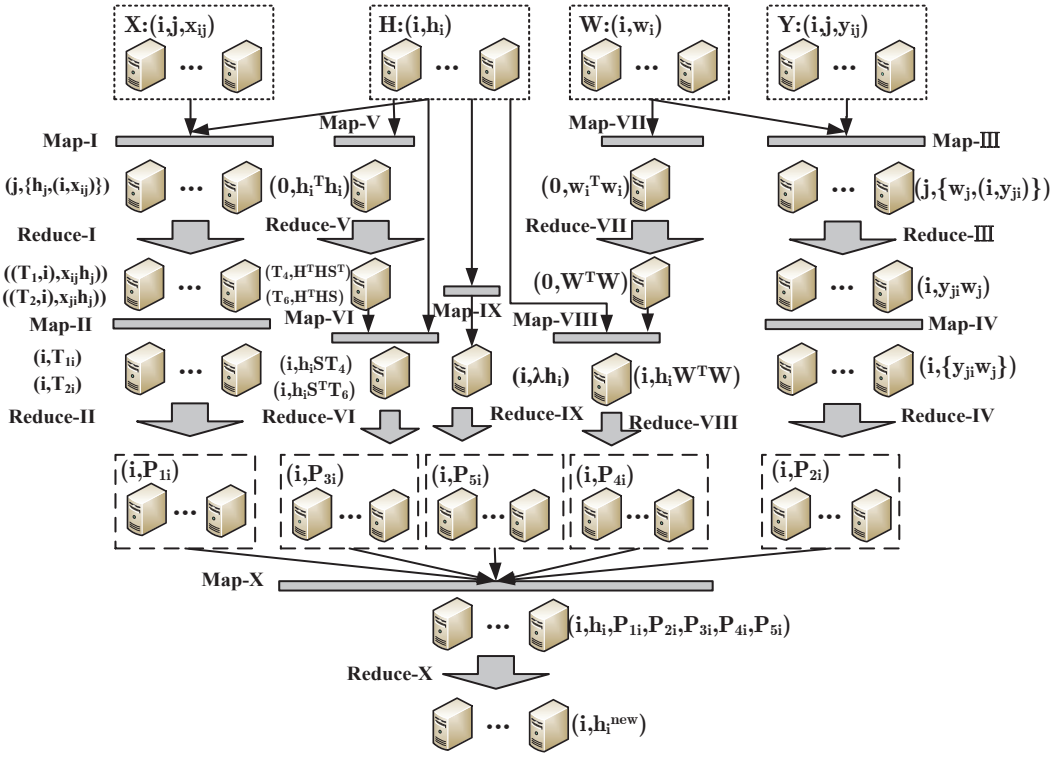

Figure 4. Computing H on MapReduce.

3.4. Theoretical modelling on α

In Equation 1, when $\alpha = 1$, our method follows traditional ways of friend recommendation using link information. When $\alpha = 0$, users are more preferably recommended by several of tags extracted from content information instead of by sharing friends connections. Here another question arises, what kind of α would be our methods initial setting? Basically, we can use a theory to support and define the initial value for α, denoted as α_0. Then the parameter can vary with respect to time, for later analysis of how important is the link information when comparing it with the tags, we can have a sequence of

$$\alpha_{iT} = \alpha_{(i-1)T} + \varepsilon_{iT} \tag{12}$$

where $i \geq 1$ and "T" just indicates that α_i and ε_i vary with respect to a certain period of time like one week or 1,000,000 counted views. Since we have introduced a static variable ε_i, such that when $\varepsilon_i > 0$, means with previous α_i, more user want result recommended more related to links, and opposite for $\varepsilon_i < 0$. One advantage of using this model is that, we can allow later data analysis of how the user finds the importance of links and tags. In advance, if we group the tags, we can then analyze what kind of tags will be affected the most and the change of the importance with respect to time period.

One possible way to set up α_0 would be using six degrees of separation theory. That is, you can find everyone in the social network within six steps. Hence, the status zero is yourself, and status one is friends you already know in your social network, so you at most need five intermediate friends. Therefore, based upon this theory, we have when $d = 1$, $\alpha = 1$; and when $d = 6$, $\alpha = 0$, where d is our degrees. Then we use the following method: use continuous function to estimate discrete degrees, and find both continuous mean and discrete mean. In order to satisfy the above condition,

DOI: 10.1002/cpe

we may use linear model to path those two points: It is not hard to find out the equation would be $\alpha = -1/5 * d + 6/5$. From this equation, we find out our continuous mean would be:

$$\alpha_0 = 1/(6-1)\int_1^6 (-1/5 * d + 6/5)\mathbf{d}d = 0.5 \tag{13}$$

In addition, we have the following discrete table:

Table II. Discrete table.

d	α
1	1
2	4/5
3	3/5
4	2/5
5	1/5
6	0

Hence, our discrete mean would be 0.5.

For the linear case, our continuous mean is equal to the discrete mean. Therefore, if we apply strong six degree of separation with linear model, our α_0 can be suggested to be 0.5. Note that our OSNs do not fully satisfy the conditions of six degrees of separation theory, we will discuss the weight parameter selection in section 4.3.2. Also, there are a lot of other theories and models to give a suggested α_0. But due to space limitation, we omit them here.

4. EXPERIMENTAL STUDY

4.1. Comparative methods and experimental datasets

We compare the performance of our proposed method (For convenience, we call it TCB hereafter) with the following methods:

- Friends-of-Friends (FoF) [35]. This is a classical link-based method which is often used in OSNs. If a particular user has many common friends with a target user, then he or she might be very interested in being friends with the target user too. Therefore, we can recommend the Top-K users with the most common friends to the target user.
- Profile-based (PB) [25]. This is a content-based method which is also often used in OSNs. Given a target user, we compute his/her pairwise profile similarities with others. Then we find the Top-K users with the highest similarities to recommend to the target user.
- RFG [28]. This is a hybrid method that considers user link and content information for recommendation. We first construct heterogeneous networks by combining features of users link and content information and then use the ranking factor graph model (RFG) to predict friend links in OSN.

We use three real world OSN datasets for our experiments. The first one is from LinkedIn[†]. The second one is from Weibo[‡], which is the biggest microblog system in China. The third one is Flixster[§], which is a popular movie review OSN. All of datasets include public friendship or followership links, tags, profiles and posts crawled from users. Table III gives the statistics of these datasets after preprocessing, where $Sparisity(X) = 1 - |E|/(|U| \times |U|)$ and $Sparisity(Y) = 1 - |\{y_{ij}|y_{ij} \in Y \wedge y_{ij} \neq 0\}|(|U| \times |A|)$. From Table III we can see that X and Y are all extremely sparse matrices.

[†] http://www.linkedin.com
[‡] http://weibo.com
[§] http://www.flixster.com/

DOI: 10.1002/cpe

Table III. Statistics of datasets.

Statistics	LinkedIn	Weibo	Flixster
$\|U\|$	183647	234832	126936
$\|E\|$	21710923	33231784	14271458
$\|A\|$	70631	94537	60315
$Sparsity(X)$	99.9%	99.9%	99.9%
$Sparsity(Y)$	99.9%	99.9%	99.9%

4.2. Evaluation metrics

To evaluate the performance of various friend recommendation methods above, we select conversion rate (CR), precision and recall suggested in [35] as our evaluation metrics. CR is a widely used metric in recommender systems to evaluate if a user has obtained at least one good recommendation. CR is a rough metric, but precision and recall are all precise metrics. If L is the list of recommended friends and L' is the list of friends actually accepted by the user, then the standard definitions for CR, precision and recall are:

$$CR = \begin{cases} 1 & \text{if } |L \bigcap L'| > 0 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

$$Precision = \frac{|L \bigcap L'|}{L} \tag{15}$$

$$Recall = \frac{|L \bigcap L'|}{L'} \tag{16}$$

In our next experiments except the scalability tests, for every dataset we randomly choose 10% friends of each user as the testing dataset, and the rest as the training dataset. We compute the CRs, Precisions and Recalls of the various methods by taking the average of values computed for each test user. Meanwhile, every experiment is repeated 5 times and selects the average of every metric value as the final result.

4.3. Parameter sensitivity experiments

In this part we examine how parameters r and α affect the performance of our proposed TCB method. Their optimal settings will be used in our comparative experiments with other methods.

4.3.1. Number of communities r analysis. The appropriate setting of r can obtain high-quality topic communities that are very useful for improving the performance of friend recommendation. To evaluate the quality of topic community mining in OSN, we use two metrics of $Density$ and $Entropy$ presented in [36]. Their definitions are as follows.

$$Density(\{c_i\}_{i=1}^{r}) = \sum_{i=1}^{r} \frac{|\{(u_p, u_q)|u_p, u_q \in c_i, e_{pq}) \in E\}|}{|E|} \tag{17}$$

$$Entropy(\{c_i\}_{i=1}^{r}) = \sum_{i=1}^{m} \sum_{j=1}^{r} \frac{|c_j|}{|U|} entropy(a_i, c_j) \tag{18}$$

where $entropy(a_i, c_j) = -p_{ij} \log_2 p_{ij}$, p_{ij} is the percentage of users in community j which have values on content feature a_i. Members in the ideal topic community not only link densely, but also share more similar content features. Therefore, the larger of $Density$ and the smaller of $Entropy$, the quality of topic community mining will be better. We fix $\alpha = 0.5$, $\lambda = 1$ for LinkedIn, Weibo and Flixster datasets, and vary the number of communities r to evaluate the quality of topic community mining and the corresponding performance of Top-K friends recommendation. The results are

DOI: 10.1002/cpe

shown in Table IV and Table V (Due to space limitations, we only show the results of Top-10 friends recommendation), respectively. From the results, we find that when we set $r = 1500$, 2000 and 1000 for LinkedIn, Weibo and Flixster datasets, respectively, they all obtain the best quality of topic community mining. Meanwhile, they also obtain the best performance of Top-K friends recommendation. The results provide evidence that high quality topic community is an important guarantee to improve friend recommendation. In the rest of the experiments, we set $r = 1500$ for the LinkedIn dataset, $r = 2000$ for the Weibo dataset and $r = 1000$ for the Flixster dataset.

Table IV. Topic community mining quality with different r.

Dataset	Metric	$r = 1000$	$r = 1500$	$r = 2000$	$r = 2500$	$r = 3000$
LinkedIn	Density	0.3236	**0.4567**	0.2987	0.3124	0.3098
	Entropy	7.5684	**5.6245**	6.4896	7.2153	8.6354
Weibo	Density	0.2965	0.3126	**0.3984**	0.3654	0.2846
	Entropy	8.1563	8.6984	**7.1236**	7.9864	9.0123
Flixster	Density	**0.4012**	0.3845	0.3624	0.3321	0.3526
	Entropy	**6.5368**	7.4587	7.8694	8.9654	9.4568

Table V. Performance of Top-10 friends recommendation with different r.

Dataset	Metric	$r = 1000$	$r = 1500$	$r = 2000$	$r = 2500$	$r = 3000$
LinkedIn	CR	0.42	**0.48**	0.41	0.39	0.38
	Precision	0.25	**0.30**	0.26	0.23	0.27
	Recall	0.26	**0.36**	0.24	0.28	0.23
Weibo	CR	0.33	0.36	**0.43**	0.35	0.37
	Precision	0.25	0.27	**0.29**	0.24	0.26
	Recall	0.23	0.26	**0.33**	0.28	0.29
Flixster	CR	**0.37**	0.31	0.36	0.33	0.32
	Precision	**0.30**	0.26	0.24	0.25	0.23
	Recall	**0.31**	0.25	0.27	0.25	0.26

4.3.2. Weight parameter α analysis. In Equation 1, the weight parameter α is used to control contributions to the final solution for link and content information. Small values of α under-fit whereas large values of α over-fit the data and this will lead to poor performance. Thus, we have to find a balance between the two kinds of information that fits best the dataset at hand. We conduct experiments on three datasets to recommend Top-2, Top-4, Top-5, Top-8 and Top-10 friends by constantly changing the value of α and find that setting α between 0.5 and 0.7 can achieve stable and high performance (Figure 5-7). The results suggest that giving slightly more importance to the link information is helpful to improve friend recommendation performance. Furthermore, we could compute the degree d is in [2.5, 3.5] based on six degree of separation linear model discussed in section 3.4. It is a reasonable distance that user discovers interested friends in the real world OSN, since most people like to add friends within 2 intermediate friends. The reason of having computed half degrees might cause by that the experiments network density is not enough to be fully using the strong six degree of separation. Therefore, one advanced research question might be: would choosing different α_0 improve the performance of friend recommendation?

From the results shown in Figure 5 to Figure 7, we can also find that TCB method still has a good recommendation performance when $\alpha = 0$. It means that cold start users without any link information still can obtain friend recommendation. Therefore, our proposed method is effective to address the problem of cold start by combing user link and content information.

DOI: 10.1002/cpe

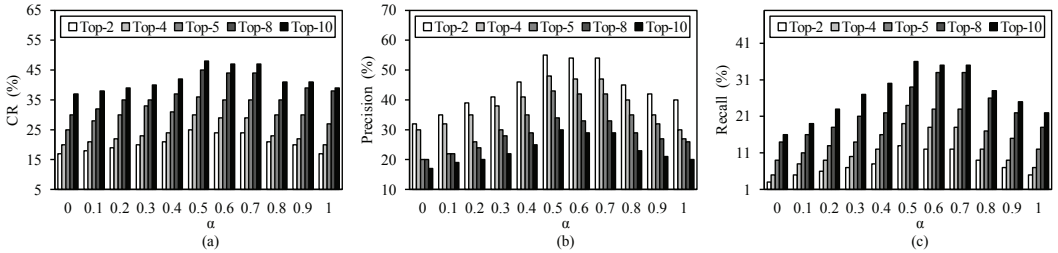

Figure 5. Performance on LinkedIn dataset with different α.

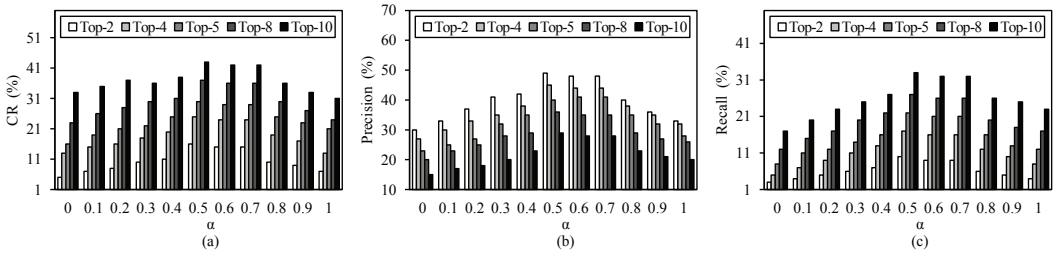

Figure 6. Performance on Weibo dataset with different α.

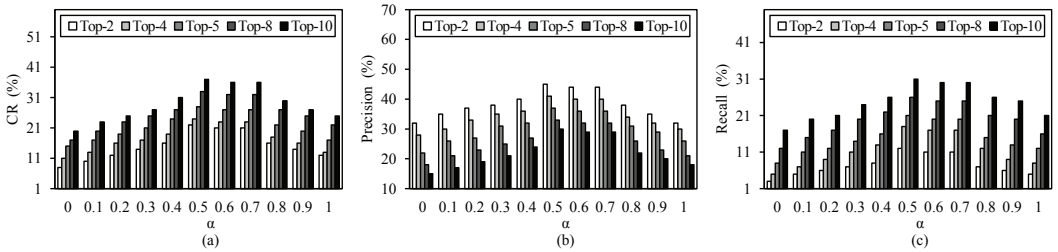

Figure 7. Performance on Flixster dataset with different α.

4.4. Comparative experiments

We first set the optimal parameters suggested in section 4.3 for LinkedIn, Weibo and Flixster dataset and then compare the performance of various methods to recommend Top-2, Top-4, Top-5, Top-8 and Top-10 friends, respectively. Figure 8 to Figure 10 show the comparison results of various evaluation metrics on these three datasets. From the results, it is clear that hybrid methods (TCB and RFG) outperform link-based (FoF) and content-based (PB) methods. This is a further verification that it is more effective to make friend recommendation by combining user link and content information. Among the two hybrid methods, our proposed TCB method performs better than RFG method. This is because RFG needs to compute user pairwise similarities based on the X and Y matrices for constructing an attribute augmented friendship network, but in Table III, X and Y on three datasets are so sparse that it is easy to produce errors in computing user similarity and will finally affect the recommendation performance of the RFG model. However, TCB computes the similarity based on the community indicator matrix H. H is a low-rank matrix and has much lower sparsity compared to X and Y. More importantly, H performs well in terms of keeping the main features of every user, so TCB still can improve the computational accuracy of similarity and the final recommendation performance under the condition of data sparsity.

4.5. Scalability test on Hadoop cluster

We select the open source cloud computing platform Hadoop to test the scalability of TCB implemented by MapReduce. Our Hadoop cluster is composed of 10 machines which have the

DOI: 10.1002/cpe

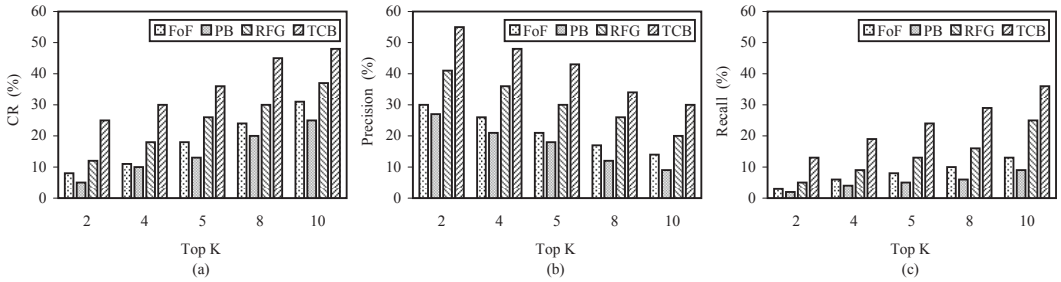

Figure 8. Comparative results on LinedIn dataset.

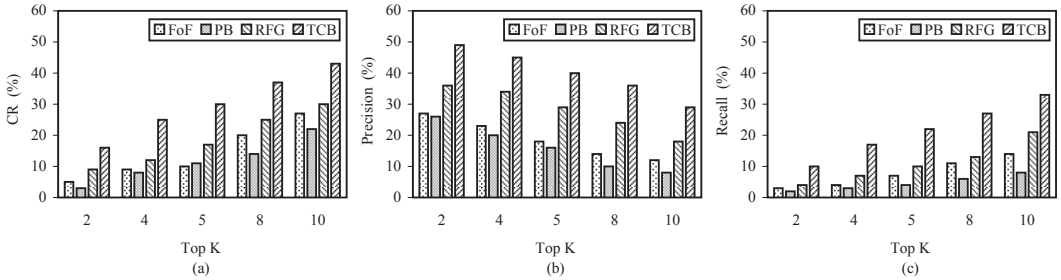

Figure 9. Comparative results on Weibo dataset.

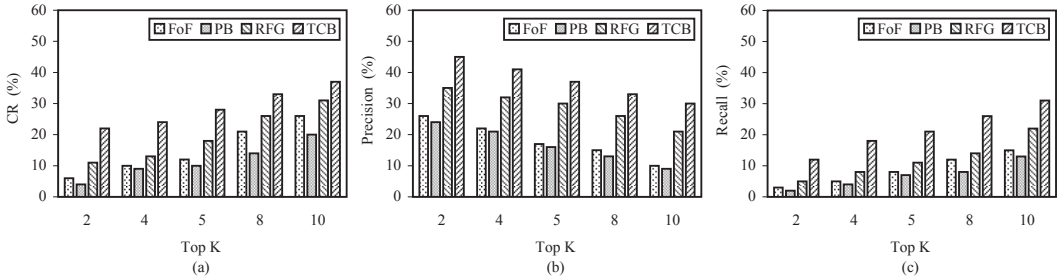

Figure 10. Comparative results on Flixster dataset.

same configuration: 2.93G Hz CPU, 8GB memory and 1TB disk. Because updating H, S and W iteratively is the most time-consuming task in TCB, we mainly test the scalability of updating H, S and W iteratively on Hadoop cluster. Firstly, we stored the corresponding X and Y of three datasets in the format of key-value pairs. In the Hadoop distributed file system, matrix file sizes of X and Y of LinkedIn dataset are 135.2MB and 21.5MB, respectively. Matrix file sizes of X and Y of Weibo dataset are 213.5MB and 28.7MB, respectively. Matrix file sizes of X and Y of Flixster dataset are 91.7MB and 18.9MB, respectively. For H, S and W, they are all dense matrices after the first initialization and are also suitable to be stored in the format of key-value pairs. Table VI lists the corresponding matrix file sizes of H, S and W of three datasets (in MB) when we set $r = 1000, 1500, 2000, 2500, 3000$, respectively. From Table VI, we can see that S matrix file size is small, so it is easy to be directly loaded into the Hadoop cluster distributed cache. However, H and W matrix file sizes are so large that they are hard to be processed using single machine due to limited CPU and memory resources. We update H, S and W iteratively on Hadoop cluster and test the scalability through changing the number of worker machines in cluster and the value of r. We introduce the $Speedup = T_s/T_c$ as the evaluation criterion, where T_s is the running-time of Hadoop clusters under standalone mode, namely only includes one worker machine, and T_c is the running-time of Hadoop cluster with specified number of worker machines [37]. The corresponding

DOI: 10.1002/cpe

Table VI. H, S and W matrices initialization file size with different r.

Dataset	Matrix file	$r = 1000$	$r = 1500$	$r = 2000$	$r = 2500$	$r = 3000$
LinkedIn	H	1722	2741	3445	4330	5212
	S	9	12	20	24	33
	W	1275	1910	2502	3190	3846
Weibo	H	2050	3118	4075	5015	6355
	S	9	12	20	24	33
	W	1497	2250	3015	3781	4568
Flixster	H	1411	2107	2615	3246	3864
	S	9	12	20	24	33
	W	1286	1903	2436	3053	3585

speedup changing curves of updating H, S and W in the first iteration on different dataset are shown in Figure 11-13, respectively.

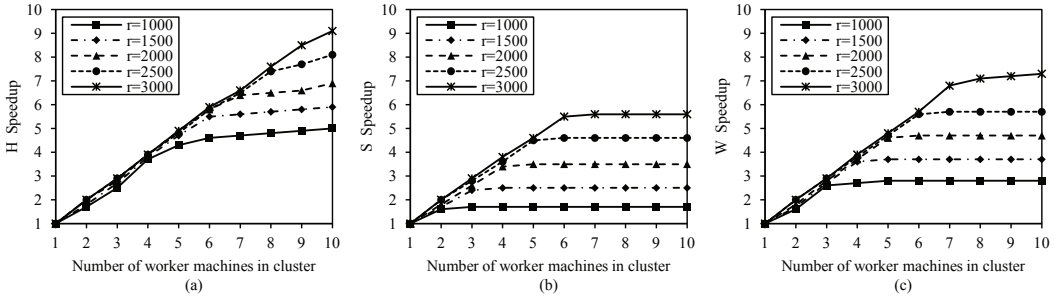

Figure 11. Speedup of updating H, S and W on LinkedIn dataset.

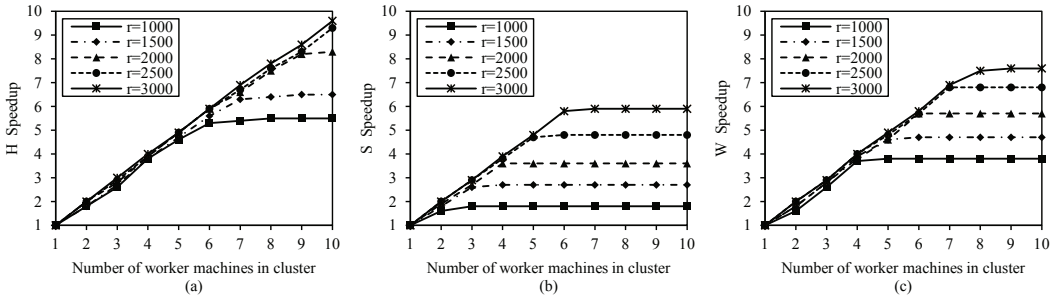

Figure 12. Speedup of updating H, S and W on Weibo dataset.

From Figure 11-13, we can see that all the speedup changing curves increase linearly with the increase in the number of worker machines in cluster. However, when the number of worker machines continues to increase to a threshold, the speedup will not grow any more, and stabilize at a certain value. The reason is that the size of dataset limits the number of MapReduce tasks. When the number of worker machines is greater than the number of tasks, the tasks cannot be fully parallelized to all worker machines in the cluster, so the speedup will no longer increase. For example, in Figure 11(b) all the speedup changing curves of updating S on LinkedIn dataset become stable after the number of worker machines increases to 6. This phenomenon can also be discovered in other Figures. Interestingly, we notice that when matrices H and W becomes more and more sparser through continuous iterative updating, their speedups actually drops in every iteration. This

DOI: 10.1002/cpe

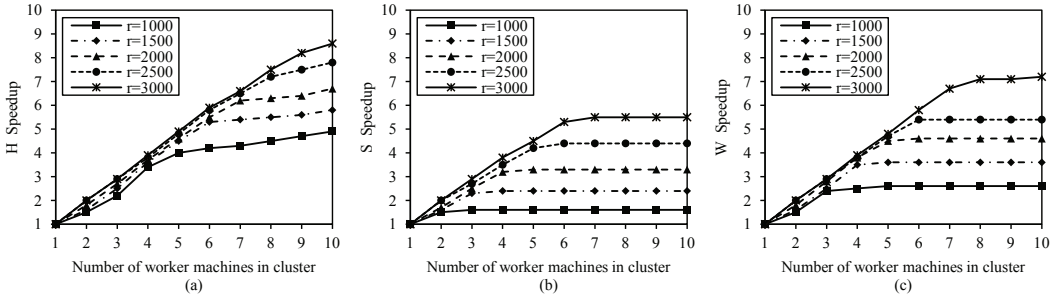

Figure 13. Speedup of updating H, S and W on Flixster dataset.

suggests that the overhead actually outweighs real computation for these small datasets that can be processed with very few worker machines. In other words, the Hadoop cluster is not yet saturated.

To further validate the scalability of TCB, we make a comparison with a baseline method named HAMA_MM presented in [34]. HAMA_MM can also use a series of MapReduce jobs to implement matrix multiplication. It directly maps every element in matrices into the matrix multiplication phase, and then it will produce large amount of intermediate data. Therefore, its implementation strategy is different from TCB. We change the number of worker machines in cluster and compare their running time of updating H, S and W in the first iteration on different dataset. For simplicity, every dataset selects the optimal parameters suggested in section 4.3. The results are shown in Figure 14-16, from which we can see that the running time of TCB is obviously less than that of HAMA_MM. That is because large amount of intermediate data in HAMA_MM increases its cost of data transferring and shuffling, but TCB is effective to control the the amount of intermediate data by mapping partitions along the short dimension of matrices into the matrix multiplication phase. Based on related experiments above, it is clear that TCB implemented by MapReduce has good scalability and is efficient enough to process the large-scale OSN datasets.

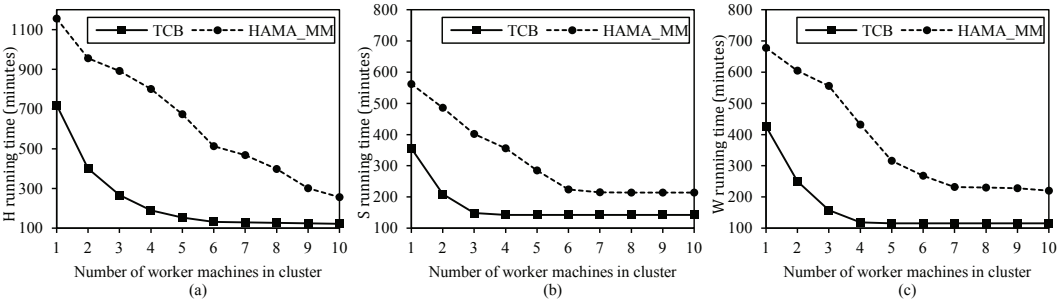

Figure 14. Running time of updating H, S and W on LinkedIn dataset.

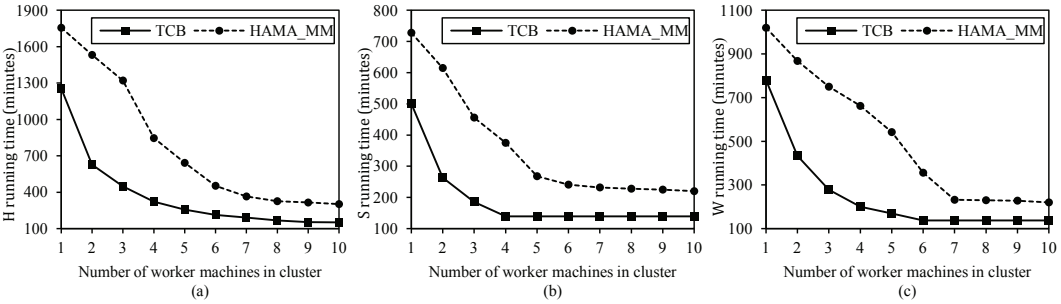

Figure 15. Running time of updating H, S and W on Weibo dataset.

DOI: 10.1002/cpe

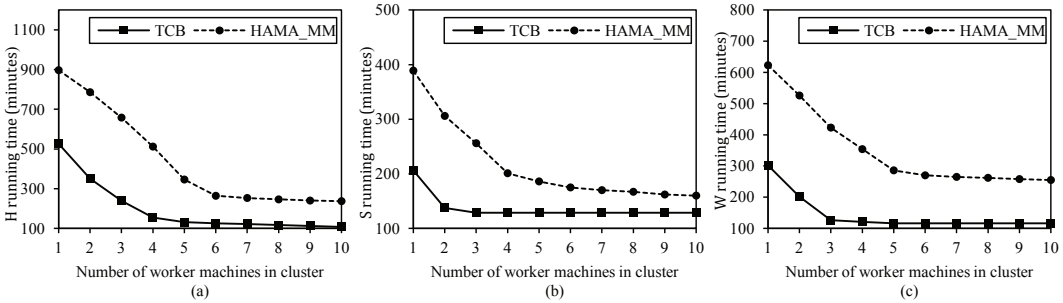

Figure 16. Running time of updating H, S and W on Flixster dataset.

4.6. Application in SCHOLAT OSN

SCHOLAT[¶] is a vertical OSN website, which is developed by our research team and mainly provides services to scientific researchers. It is very useful to help users to publish, share and obtain academic information that they are interested in. Recently, SCHOLAT has attracted over one hundred thousand registered users and has become a very promising scientific researchers OSN in China. We developed the corresponding friend recommendation system based on TCB method to provide friend recommendation service in SCHOLAT OSN. Its architecture is shown in Figure 17. Actually, the recommendation system comprises offline data processing and online recommendation service. The Data_Extrator module periodically extracts various user data from SCHOLAT OSN database, including profiles, academic publications and friendship links. After data preprocessing, the link and content feature matrix files will be directly stored in Hadoop distributed file system. Topic community mining and user similarity computing modules, which are respectively based on Algorithm 1 and Algorithm 2, can process these large-scale offline data in Hadoop distributed platform. The Usersim_Loader module will load the final results into SCHOLAT OSN database to provide users with online friend recommendation service. By analyzing user feedback data within a certain period of time, we find that our recommendation system is very effective in improving friend recommendation quality. For example, every member in a typical social network topic community (Figure 18) has high probability to accept friend recommendation from the same community. Characteristics of this community before and after recommendation are shown in Table VII, respectively. It is obvious that links become more dense in this community. Members in this community not only link more densely but also share more similar research interests. Therefore, they are more likely to become friends with each other.

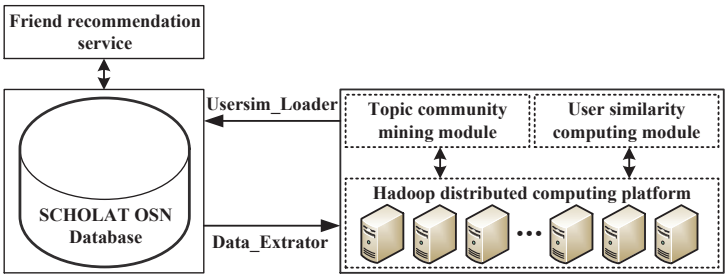

Figure 17. Architecture of friend recommendation system in SCHOLAT OSN.

[¶] http://www.scholat.com

DOI: 10.1002/cpe

Figure 18. Visualization of the social network topic community in SCHOLAT OSN.

Table VII. Characteristics of social network topic community.

Status	Avg(degree)	Density	Cluster coefficient
Before recommendation	5.123	0.007	0.246
After recommendation	11.88	0.017	0.596

5. CONCLUSIONS AND FUTURE WORK

In this paper, we study the problem of friend recommendation in OSN and propose a topic community-based method. Our method firstly mines topic communities via joint NMF model and then makes friend recommendation based on the topic communities. Experimental and application results demonstrate its effectiveness and its ability to improve friend recommendation service in the real-world OSNs. Furthermore, our method can be implemented using MapReduce and hence has good scalability to process large-scale OSN datasets. Through comparison with other representative methods we find that: (1) Combining user link and content information for recommendation is more appropriate to reflect user preferences on friend selection. (2) Mining topic communities enables us to reduce data sparsity and improve recommendation performance. Actually, our proposed method provides a good solution to three classic problems existing in friend recommendation in large-scale OSNs effectively: cold start, data sparsity and scalability. As future work, we will continue to study more strategies to improve the friend recommendation performance in OSNs and compare with more existing methods to further demonstrate the advantages of our method. Especially we have noticed that matrix X describes only the direct links, i.e., $x_{ij} = 0$ if user i and user j are not directly linked. However, user i and user j may be some degree of separation and this is not described in X. We will extend matrix X by, e.g., assigning $x_{ij} = (1/2)^n$ if user i and user j are n degree of separation. We will investigate whether this way of representing user links will achieve higher performance.

DOI: 10.1002/cpe

ACKNOWLEDGMENTS

This work was partially supported by the following projects: National High-Technology Research and Development Program ("863" Program) of China (No. 2013AA01A212), National Natural Science Foundation of China (Nos. 61370178, 61370229), Natural Science Foundation of Guangdong Province, China (Nos. S2012030006242, 2015A030310509), Science and Technology Support Program of Guangdong Province, China (Nos. 2016A030303058, 2015A020209178, 2015B010129009, 2016A090922008, 2015B010109003, 2015B010129009, 2014B010103004, 2014B010117007, 2015A030401087, 2015B010110002), Open Research Fund from Guangzhou Key Lab on Cloud Security and Assessment (No. GZCSKL-1407).

REFERENCES

1. Silva NB, Tsang IR, George DC. A graph-based friend recommendation system using genetic algorithm. *Proceedings of 6th IEEE World Congress on Computational Intelligence*, Barcelona, Spain, 2010; 233-239.
2. Xie X. Potential friend recommendation in online social network. *Proceedings of 3rd IEEE/ACM International Conference on Cyber, Physical and Social Computing*, Hangzhou, China, 2010; 831-835.
3. Naruchitparames J, Gunes MH, Lou SJ. Friend recommendations in social networks using genetic algorithms and network topology. *Proceedings of 2011 IEEE Congress on Evolutionary Computation*, New Orleans, USA, 2011; 2207-2214.
4. Tang Q, Hartel P. Poster: privacy-preserving profile similarity computation in online social networks. *Proceedings of the 18th ACM conference on Computer and Communications Security*, Chicago, USA, 2011; 793-796.
5. Samanthula BK, Wei J. Interest-driven private friend recommendation. *Knowledge and Information Systems*, 2015; 42(3):663-687.
6. Gou L, You F, Guo J, Wu L, Zhang XL. Sfviz: interest-based friends exploration and recommendation in social networks. *Proceedings of 2011 International Symposium on Visual Information Communication*, ACM, HongKong, China, 2011; 1-10.
7. McPherson M, Smithlovin L, Cook JM. Birds of a feather: homophily in social networks. *Annual Review of Sociology*, 2001; 27:415-444.
8. Zhang X, Cheng J, Yuan T. TopRec: domain-specific recommendation through community topic mining in social network. *Proceedings of the 22nd International Conference on World Wide Web*, Rio de Janeiro, Brazil, 2013; 1501-1510.
9. Lee DD, Seung HS. Learning the parts of objects by non-negative matrix factorization. *Nature*, 1999; 401(10):788-791.
10. Ding C, HE XF, Simon HD. On the equivalence of nonnegative matrix factorization and spectral clustering. *Proceedings of the 5th SIAM International Conference on Data Mining*, Philadelphia, USA, 2005; 606-610.
11. Guo YC, Ding GG, Zhou JL. Robust and discriminative concept factorization for image representation. *Proceedings of the 5th ACM International Conference on Multimedia Retrieval*, Shanhai, China, 2015; 115-122.
12. Long XZ, Lu HT, Peng Y. Graph regularized discriminative non-negative matrix factorization for face recognition. *Multimedia Tools and Applications*, 2014; 72(3):2679-2699.
13. Xu W, Liu X, Gong YH. Document clustering based on non-negative matrix factorization. *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, New York, USA, 2003; 267-273.
14. Kannan R, Ishteva M, Park H. Bounded matrix factorization for recommender system. *Knowledge and Information Systems*, 2014; 39(3):491-511.
15. Wu ZA, Cao J, Wu JJ, Wang YQ, Liu CY. Detecting genuine communities from large-scale social networks-a pattern-based method. *The Computer Journal*, 2014; 57(9):1143-1157.
16. Wang F, Li T, Wang X. Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery*, 2011; 22(3):493-521.
17. Psorakis I, Roberts S, Ebden M. Overlapping community detection using bayesian non-negative matrix factorization. *Physical Review E*, 2011, 83(6):1-9.
18. Zhang Y, Yeung DY. Overlapping community detection via bounded nonnegative matrix tri-factorization. *Proceedings of the 18th ACM International Conference on Knowledge Discovery and Data Mining*, Beijing, China, 2012; 606-614.
19. Lin YR, Sun J, Castro P. Metafac: community discovery via relational hypergraph factorization. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, 2009; 527-536.
20. Zhang ZY. Community structure detection in social networks based on dictionary learning. *Science China Information Sciences*, 2013, 56(7):1-12.
21. Zhang ZY, Wang Y, Ahn YY. An overlapping community detection in complex networks using symmetric binary matrix factorization. *Physical Review E*, 2013, 87(6):062803.
22. Zhu J, Xie Q, Chin EJ. A hybrid time-series link prediction framework for large social network. *Lecture Notes in Computer Science*, 2012; 7447:345-359.
23. Armentano MG, Godoy D, Amandi A. Topology-based recommendation of users in micro-blogging communities. *Journal of Computer Science and Technology*, 2012; 27(3):624-634.

DOI: 10.1002/cpe

24. Carullo G, Castiglione A, Santis AD, Palmieri F. A triadic closure and homophily-based recommendation system for online social networks. *World Wide Web*, 2015; 18(6):1579-1601.
25. Akcora CG, Carminati B, Ferrari E. User similarities on social networks. *Social Network Analysis and Mining*, 2013; 3(3):475-495.
26. Wan SX, Lan YY, Guo JF. Informational friend recommendation in social media. *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Dublin, Ireland, 2013; 1045-1048.
27. Wang ZB, Liao JL, Cao Q. Friendbook: a semantic-based friend recommendation system for social networks. *IEEE Transactions on Mobile Computing*, 2014; 14(3):538-551.
28. Dong YX, Tang J, Wu S. Link prediction and recommendation across heterogeneous scial networks. *Proceedings of the 12th IEEE International Conference on Data Mining*, Brussels, Belgium, 2012; 181-190.
29. Cummins R, Riordan CQ. An axiomatic comparison of learned term-weighting schemes in information retrieval: clarifications and extensions. *Journal of Artificial Intelligence Review*, 2007; 28(1):51-68.
30. Guo G, Zhang J, Thalmann D. Merging trust in collaborative filtering to alleviate data sparsity and cold start. *Knowledge-Based Systems*, 2014; 57(2):57-68.
31. Hu Y, Peng Q, Hu X, Yang R. Time aware and data sparsity tolerant web service recommendation based on improved collaborative filtering. *IEEE Transactions on Services Computing*, 2015; 8(5):782-794.
32. Liu C, Yang HC, Fan J, He LW, Wang YM. Distributed nonnegative matrix factorization for web-scale dyadic data analysis on MapReduce. *Proceedings of the 19th International Conference on World Wide Web*, Raleigh, USA, 2010; 681-690.
33. Sindhwani V, Ghoting A. Large-scale distributed non-negative sparse coding and sparse dictionary learning. *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Beijing, China, 2012; 489-497.
34. Seo S, Yoon EJ, Kim J, Jin S, Kim JS, Maeng S. HAMA: An Efficient Matrix Computation with the MapReduce Framework. *Proceedings of the 2nd International Conference on Cloud Computing Technology and Science*, Indianapolis, USA, 2010; 721-726.
35. Zhao G, Li M, Hsu W, Chen W, Hu HJ. Community-based user recommendation in uni-directional social networks. *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, San Francisco, USA, 2013; 189-198.
36. Cheng H, Zhou Y, Yu JX. Clustering large attributed graphs: a balance between structural and attribute similarities. *ACM Transactions on Knowledge Discovery from Data*, 2011; 5(2):190-205.
37. He CB, Tang Y, Yang ZX, Zheng K, Chen GH. SRSH: A social recommender system based on Hadoop. *International Journal of Multimedia and Ubiquitous Engineering*, 2014; 9(6):141-152.

DOI: 10.1002/cpe