International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.6 (2014), pp.141-152
http://dx.doi.org/10.14257/ijmue.2014.9.6.14

SRSH: A Social Recommender System based on Hadoop

Chaobo He[1, 2], Yong Tang[2*], Zhenxiong Yang[2], Kai Zheng[2] and Guohua Chen[2]

[1] *School of Information Science and Technology, ZhongKai University of Agriculture and Engineering, Guangzhou 510225, China*
[2] *School of Computer, South China Normal University, Guangzhou 510631, China*

hechaobo@foxmail.com, ytang@m.scnu.edu.cn, toyangzx@163.com, david@scnu.edu.cn, chguohua@gmail.com

Abstract

Online Social Networks (OSNs) accumulate a large amount of user-generated data and Social Recommender Systems (SRSs) can help users discover information they are interested. However, most of the existing SRSs do not have good scalabilities to process huge volumes of data. Aiming to this problem we design a social recommender system named SRSH, which is based on Hadoop parallel computing platform. SRSH provides second-degree friends, similar users, user community and content recommendation modules, which can meet user needs of finding potential friends and attractive content. Especially, every core methods existing in these modules above can be implemented using MapReduce parallel programming framework and run in Hadoop cluster. We have conducted extensive related experiments on the realistic dataset and the experimental results show that SRSH scales well and has the ability of dealing with the problem of recommendation in the large-scale OSN.

Keywords: *Social recommender system, Hadoop, MapReduce*

1. Introduction

Online Social Networks (OSNs), such as Facebook, Tweeter and LinkedIn, have become tremendously popular in recent years. Millions of users are active daily in these sites and create a large amount of data online that has not been available before, including user links data and other user-generated content, such as blogs, photos, videos, *etc.* However, the abundance and popularity of OSNs flood users with huge volumes of information and hence bring users the problem of information overload. For example, the specific user is difficult to find potential friends he wants to know and is hard to discover content he wants to consume. Aiming to alleviate information overload over OSNs users, Social Recommender System (SRS) is proposed. This kind of system is different from traditional recommender systems using content-based methods or collaborative filtering methods separately and it can incorporate new techniques that take advantage of explicit links information between users in OSNs to make more effective recommendation. Recently, many researchers have paid close attention to SRS and tried to improve the performance of SRS from different perspectives. J. M. He *et al.* [1] presented a recommender system which can utilize influence information from social friends. This system used a probabilistic model to make personalized recommendations and performed well on the realistic OSNs dataset. E. Davoodi *et al.* [2] developed an expert recommendation system that integrated the characteristics of

ISSN: 1975-0080 IJMUE

content-based recommendation algorithms into a social network-based collaborative filtering system. Experimental results showed that this system had a good recommendation precision. M. S. Pera *et al.* [3] presented a personalized book recommendation system named PBRecS, which utilized social interactions and personal interests to suggest books appealing to users. PBRecS relied on the friendships established on a social networking site and can generate more personalized suggestions. J. Golbeck *et al.* [4] used a social network approach to develop FilmTrust system which effectively recommended movies to the target user based on ratings of his friends and his trust weights to friends. Although these existing SRSs above have been proved that they have better recommendation precision than traditional recommender systems which do not consider social information between users, they all face the problem of poor scalability. Realistic OSNs often possess a large amount of data, including the large-scale complex social graph data and user-generated content. All of these need more effective computing method than before. If SRS does not scale well enough to process large datasets existing in OSNs, the performance of SRS will be poor. Aiming at this problem, we propose to design a social recommender system named SRSH, which is based on Hadoop, to provide a perfect scalable solution to SRS. Our main works include three aspects shown as follows:

(1) We design a social recommender system named SRSH, which comprises four core modules, *i.e.*, second-degree friends recommendation module, similar users recommendation module, user community recommendation module and content recommendation module. In particular, we also integrate content-based and collaborative filtering techniques into our system to further improve the performance of recommendation.

(2) Using Hadoop, we implement core algorithms of SRSH, such as the algorithm of potential friends discovery in social graph, the algorithm of pairwise user similarity computation and the algorithm of content recommendation using collaborative filtering.

(3) We conduct extensive experiments on a realistic OSN dataset. Experimental results show that SRSH has a good recommendation precision and especially scales well to have the ability of solving the problem of personalize recommendation in the large-scale OSN.

The rest of this paper is organized as follows. Section 2 reviews related work, including the introduction of Hadoop platform and the MapReduce distributed programming model. In section 3 we present the system architecture of SRSH. Details about core algorithms implemented within MapReduce are presented in section 4. Section 5 shows the experimental results and evaluations. In Section 6 we draw the final conclusions and outline the future work.

2. Related Work

Hadoop is an open-source platform that allows for the distributed processing of large datasets across clusters of computers using simple programming model. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Hadoop comprises two important modules: HDFS and MapReduce. HDFS is a distributed file

system that provides high-throughput access to application data and MapReduce is a programming model for parallel processing of large datasets. It is notable that applications based on Hadoop should be implemented using MapReduce [5-6]. MapReduce builds on the observation that many tasks have the same structure: a computation is applied over a large number of records (*e.g.*, documents) to generate partial results, which are then aggregated in some fashion. Taking inspiration from higher-order functions in functional programming, MapReduce provides an abstraction that involves the programmer defining a "mapper" and a "reducer", with the following signatures:

$$\text{Map}:(in\_key, in\_value) \rightarrow \{(key_i, value_i) \mid i = 1...k\}$$
$$\text{Reduce}:(key_i, [v_1, ..., v_m]) \rightarrow (key_i, out\_value)$$

Key/value pairs form the basic data structure in MapReduce. The "mapper" is applied to every input key/value pair to generate an arbitrary number of intermediate key/value pairs. The "reducer" is applied to all values associated with the same intermediate key to generate output key/value pairs.

Recently, there have been a lot of studies on processing large-scale datasets using Hadoop and MapReduce, especially in the aspect of OSN analysis and mining. K. Shuang *et al.* [7] proposed X-RIME: a Hadoop-based analysis tool for large-scale social network. It was developed by MapReduce programming model and run on Hadoop. Its performance is evaluated with experiments, which demonstrates its good scalability. G. J. Liu *et al.* [8] used Hadoop and MapReduce to conduct a series of analyses on large-scale social networks including several distributions, clustering coefficient and diameter. Extensive experimental results shown that Hadoop can solve large-scale social networks analysis problem by making use of the power of multi-machines. P. Sharma *et al.* [9] presented strategies for speeding up calculation of social network graph metrics and layout by exploiting the parallel architecture of Hadoop platform. They also designed a social network analysis tool that was faster and more scalable than before. J. Tang et al. [10] proposed the topic-level social influence algorithm which was designed with efficient distributed learning algorithms that was implemented and tested under the MapReduce framework. Related experiments on real large datasets demonstrated its effectiveness and efficiency. From these studies above we can learn that if core algorithms in SRS can be implemented using MapReduce, it is feasible to solve the problem of scalability existing in SRS based on Hadoop. Accordingly, in this paper we dedicate to implement every core algorithm existing in SRSH using MapReduce and demonstrate the implementation details.

3. System Overview

SRSH provides users with necessary services which are composed of potential friends recommendation and content recommendation. Potential friends recommendation can help the target user to find users he is interested in and extend his social network graph by making friends. For improving the acceptance rate of friends recommendation, we adopt three kinds of friends recommendation strategies, including second-degree friends recommendation, similar users recommendation and user community recommendation. Second-degree friends

mean friends of friends and are those whom the target user probably knows. They can be computed using user social network graph. Similar users are those who are very similar to the target user. They need to be computed the pairwise similarities with the target user. In SRSH, we use the similarities of user profiles to represent the similarities between users. User community recommendation is different from above and it can recommend users cluster, in which users connect with each other densely, to the target user. It needs to conduct users cluster task on user social network graph. User-generated content in OSN is abundant, such as blogs, photos, videos, etc. These kinds of content are hard to directly analyze their internal features to make recommendation and we have to consider their external information, such as ratings from users. Therefore, we can use collaborative filtering method to make content recommendation to the target user. SRSH comprises core modules corresponding to recommendation services above and every module can store data in HDFS and process data using MapReduce. HDFS and MapReduce services are both provided by Hadoop platform.

4. Core algorithms implemented using MapReduce

4.1 Second-degree Friends Recommendation (SDFR)

Second-degree friends recommendation is based on the famous theory of six degrees of separation [11], which means everyone is six or fewer steps away, by way of introduction, from any other person in the world. In realistic OSN, second-degree friends are those users whom the target user can reach to using two steps through his social network graph and the target user knows them with high possibility. For example, in Figure 1 second-degree friends of user u_3 include u_4, u_6 and u_7. Furthermore, u_4 directly connects two friends u_2 and u_5 of u_3, but u_6 and u_7 respectively connect only one friend of u_3. Therefore, u_4 can rank first in the second-degree friends recommendation list for u_3. In summary, in order to make second-degree friends recommendation to the target user we can first produce candidates from friends of his friends and then remove the first-degree friends from candidates. Finally, we can rank the rest of candidates using support metric which means the more first-degree friends connect one candidate, the better rank for this candidate. Let the whole social network graph data store in the format of $< uida, uidb >$ records which represent friendship links between users. Then, the second-degree friends recommendation algorithm can be implemented using two separate MapReduce jobs illustrated in Algorithm 1.

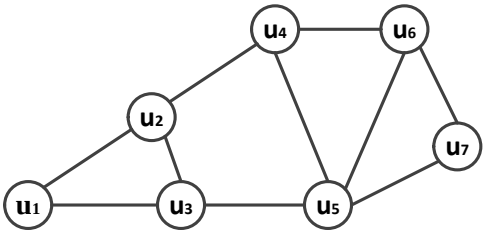

Figure 1. Social Network Subgraph

Algorithm 1: Second-degree friends recommendation using MapReduce

```
//MapReduce job for producing candidate second-degree friends
Map(Key uid_i, Value uid_j)
if uid_i.compareTo(uid_j) < 0 Then
  Emit(uid_i, uid_j);
else
  Emit(uid_j, uid_i);
Reduce(Key uid_i, ValueIterator friendslist)
foreach uid_j ∈ friendslist do
  Emit((uid_i, uid_j), '1stdegree');
foreach uid_m ∈ friendslist do
   foreach uid_n ∈ friendslist do
    if uid_m.compareTo(uid_n) < 0 Then
      Emit((uid_m, uid_n), '2stdegree');
//MapReduce job for identifying second-degree friends
Map(Key (uid_i, uid_j), Value flag)
Emit((uid_i, uid_j), flag);
Reduce(Key (uid_i, uid_j), ValueIterator flagslist)
support = 0;
foreach flag_i ∈ flagslist do
  if flag_i.equals('1stdegree') Then return;
  else support = support + 1;
Emit((uid_i, uid_j), support);
```

4.2. Similar Users Recommendation (SUR)

In OSN user profile document often includes user demographic information, personal interests, education backgrounds, work experiences, etc. and has the stamp of the truth. If two user profile documents are very similar, then they also have the high similarity and can be recommended to each other to become friends. Accordingly, to effectively make similar friends recommendation, we need to compute pairwise similarities of user profile documents. If two users have high similarity of corresponding user profile documents, then they can be recommended to each other. We adopt a simple metric to compute document similarity, where the similarity score can be expressed as an inner product of term weights. Under "a bag of words" model, document d_i of user i is represented as a vector of term weights w_{td_i}, which can be denoted as the frequency of term t. The similarity score between two documents d_i and d_j is computed as: $sim(d_i, d_j) = \sum_{t \in V} w_{td_i} \cdot w_{td_j}$, where V is the vocabulary. Since a term will contribute to the similarity score only if it has non-zero weights in both documents, $t \in V$ can be replaced with $t \in d_i \bigcap d_j$ above. We propose an efficient solution to the massive pairwise document similarities problem, expressed as two separate MapReduce jobs:

(1) Indexing: This job is to build a standard inverted index [12], where each term associates with a list of postings which contain the *docids* and the corresponding term weights. Mapping all of user profile documents, for each term in the document the mapper emits the term as the key and the *docid* that contains it as the value. The shuffle operation can automatically group these records above and provide its results to the reducer, which finally outputs inverted index postings to disks after computing term weights.

(2) Pairwise Similarity: Mapping all of postings in the inverted index, for each posting the mapper emits the $docid$ pair as the key and their product of the corresponding term weights as the value. Especially the total number of $docid$ pairs is $m(m-1)/2$, where m is the length of the posting. The shuffle operation sorts the output of mapper and then the reducer sums all the individual score contributions for a pair to generate the final similarity score. The detail of similar users recommendation using MapReduce is shown in Algorithm 2.

Algorithm 2: Similar users recommendation using MapReduce

```
//MapReduce job for indexing
Map(Key docid, Value termslist)
foreach t_i ∈ termslist do
  Emit(t_i, docid);
Reduce(Key t_i, ValueIterator docidslist)
ht = new HashTable(); posting ← null;
foreach docid_i ∈ docidslist do
  ht(docid_i).value = ht(docid_i).value + 1;
Emit(t_i, ht);
//MapReduce job for computing pairwise similarity
Map(Key t_i, Value postinglist)
foreach p_m ∈ postinglist do
   foreach p_n ∈ postinglist do
    if p_m.docid.compareTo(p_n.docid) < 0 Then
      Emit((p_m.docid, p_n.docid), p_m.wt * p_n.wt);
Reduce(Key (docid_i, docid_j), ValueIterator productlist)
similarity = 0;
foreach p_i ∈ productlist do
  similarity = similarity + p_i;
Emit((docid_i, docid_j), similarity);
```

4.3. User Community Recommendation (UCR)

In OSN users who have the similar interests often connect to each other closely and hence OSN will form many different user groups, which are also called user communities. Finding a community in OSN is to identify a set of users such that they interact with each other more frequently than with those users outside the group. If we can discover communities existing in OSN, then we can recommend the most interesting community to the target user, to which he belongs to. Recently many community mining approaches have been proposed. These approaches can be divided into four categories: node-centric, group-centric, network-centric and hierarchy-centric [13]. For practical use with the large-scale OSN, these approaches must have good scalabilities. In this paper we use classical K-means clustering algorithm implemented using MapReduce to mine community in the large-scale OSN. Let $G=(V,E)$ denotes the OSN, where the set of nodes V corresponds to users and the set of edges E corresponds to links between users in OSN, respectively. The number of nodes in the OSN is n and the matrix $A \in \{0,1\}^{n \times n}$ represents the adjacency matrix of the OSN. An entry $A_{ij} \in \{0,1\}$ denotes if there is a link between nodes v_i and v_j. The similarity of node v_i and v_j can be computed using cosine similarity of their corresponding row vectors in A, namely,

$$sim(v_i, v_j) = \cos(\theta) = \frac{A_i \cdot A_j}{\|A_i\| \|A_j\|} \quad (1)$$

where A_i and A_j respectively denote the row vectors of node v_i and v_j. Clustering nodes using K-means is an iterative procedure. It first randomly selects K nodes as the initial centroids, and then iteratively forms K clusters by assigning all nodes to the closest centroid and computes the new centroid of each cluster until these centroids do not change. Every iteration in clustering executes one round of MapReduce job, the detail of which is illustrated in Algorithm 3.

Algorithm 3: User community mining using MapReduce

```
//MapReduce job for clustering users
Map(Key uid_i, Value vector_i)
clist = Getclusterinfo();
foreach c_j ∈ clist do
   simlist.add(c_j.cid, sim(vector_i, c_j.centroid));
cid = Getclosestcluster(simlist);
Emit(cid, uid_i);
Reduce(Key cid_i, ValueIterator uidslist)
centroid_i = newcentroid(uidslist);
Emit(cid_i, centroid_i);
```

4.4. Content Recommendation (CR)

In OSN user-generated content often follows lots of user evaluations, such as user ratings. Therefore, collaborative filtering technique, which is widely used in recommender system, can be utilized to recommend user-generated content to users. Collaborative filtering technique has two main categories: user-based methods [14-15] and item-based methods [16-17]. User-generated content can also be regarded as the item and in this paper we select the latter to make content recommendation. We first create the item co-occurrence matrix C and user ratings matrix U, then for the target user i we can multiple C by his rating vector U_i to get the candidate recommendation set R for him. The formalized description of our method is shown as below. Let the 3-tuple $< uid, cid, rating >$ denotes the user rating record, where uid, cid and $rating$ denote the user, item and rating, respectively. The number of items is m and the number of users is n. Then $C = \{c_{ij}\}^{m \times m}$, where c_{ij} denotes the co-occurrence times of item i and j among the user ratings records. Especially, when $i = j$, we set $c_{ij} = 0$. $U = \{u_{pq}\}^{m \times n}$, where u_{pq} denotes the rating submitted from user q to item p. For user q, his rating vector $U_q =< u_{1q}, ..., u_{mq} >^T$ and the candidate recommendation set for him is $R = C \times U_q = \{r_{s1}\}^{m \times 1}$, where $r_{s1} = \sum_{k=1}^{m} c_{sk} u_{kq}$. Let $s' = \arg\max_s r_{s1}$, if the target user does not have given rating to item s', then we can preferentially recommend item s' to him. The theory hidden in our method is that if items, to which the target user gives high ratings, have too many co-occurrence times with item s', then the value of $r_{s'1}$ will become bigger and moreover it means item s' is similar to those items which the target user prefers. Therefore, recommending item s' to the

target user is reasonable. For a given target user q, the method for making content recommendation for him includes three main steps: creating matrix C ,U and producing candidate recommendation set. Creating matrix U can be implemented simply using MapReduce and in the following Algorithm 4 we illustrate the other MapReduce jobs for creating matrix C and producing candidate recommendation set for the target user.

Algorithm 4: Content recommendation using MapReduce

```
//MapReduce job for creating Matrix C
Map(Key rid, Value <uid,cid,rating>)
Emit(uid,cid);
Reduce(Key uid, ValueIterator cidslist)
foreach cid_m ∈ cidslist do
   foreach cid_n ∈ cidslist do
    if cid_m.compareTo(cid_n) < 0 Then
      Emit((cid_m,cid_n),1);
Map(Key (cid_m,cid_n), Value times)
Emit((cid_m,cid_n),times);
Reduce(Key (cid_m,cid_n), ValueIterator timeslist)
Emit((cid_m,cid_n),timeslist.length);
//MapReduce job for producing candidate recommendation set for user q
Map(Key (cid_m,cid_n), Value times)
Emit((cid_m,cid_n),times);
Reduce(Key (cid_m,cid_n), Value times)
product = u_nq × times;
Emit(m,product);
Map(Key m, Value product)
Emit(m,product);
Reduce(Key m, ValueIterator productslist)
score = 0;
foreach p_i ∈ productslist do
 score = score + p_i;
if u_mq != null Then
Emit(m,score);
```

5. Experimental Results and Analysis

In order to test the scalability of every core module in SRSH we select the flixster dataset [18] as our experimental dataset. Flixster [19] is a social networking site for movie fans. Users can create their own profiles, invite friends, rate movies and actors, and post movie reviews as well. Flixster dataset contains user profiles, friendship links and ratings data and its detailed description of characteristics is shown in Table 1.

Table 1. Characteristics of Flixster Dataset

#Users	#links	Average links	#Items	#Ratings	Average ratings	Profiles size
786936	7058819	17.9	48794	8196077	168	2.5 GB

Our Hadoop computer cluster is composed of 6 machines, including 6 PCs. We configure one of them as the master and the others as slavers. Each PC has one processor running at

2.93GHz, 4GB memory, and 1 TB disk. We conduct two types of experiment, one is comparing the running-time on full dataset between different number of Hadoop cluster machines and the other is comparing the running-time between different ratios of dataset when the number of Hadoop cluster machines is set to 6. We introduce the speedup ratio (sr) as the evaluation criterion which is defined as: $sr = T_c / T_s$, where T_c is the running-time of Hadoop cluster with assigned number of machines and T_s is the running-time of Hadoop platform under standalone model, namely only incudes the server. In the first experiment we increase the number of PCs in Hadoop cluster from 3 to 6 and the corresponding speedup variable curves of every core module are shown in Figure 2. In the second experiment we sample the dataset into subsets of 30, 60, 80 and 100 percent of the dataset and the running-time variable curves of every core module are shown in Figure 3.

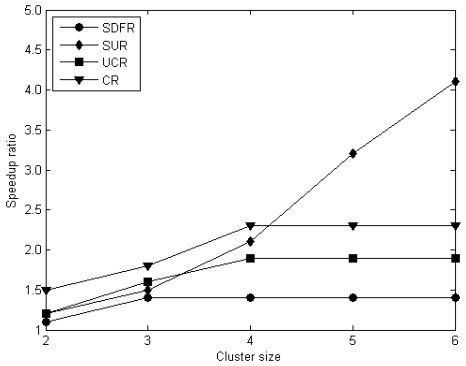

Figure 2. Speedup variable curves on different size of cluster

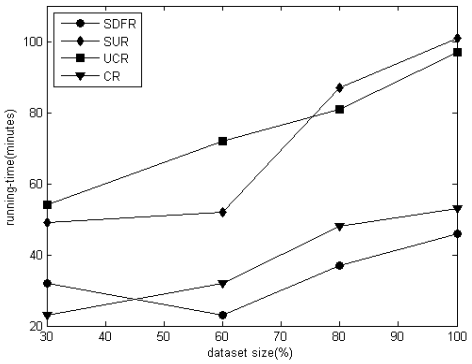

Figure 3. Running-time variable curves on different ratio of dataset

From Figure 2 we can see that all the speedup variable curves increase linearly with the increase in the size of Hadoop cluster. But when the size of cluster 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 size of Hadoop cluster is greater than the number of tasks, the tasks cannot be fully parallelized to all nodes in the cluster, so the speedup will no longer increase. For example SDFR, UCR and CR speedup variable curves become stable after the number of Hadoop cluster increase to 4. This phenomenon can also be discovered in Figure 3. With the increase of ratio of dataset the running-time of every core module does not increase linearly. On the contrary, some modules take less time on high ratio of dataset than that on low ratio of dataset. For example SDFR module takes less time on 60% dataset than that on 30% dataset. In summary, related experiments above have proved that SRSH has good scalability to process the large-scale OSN dataset.

6. Conclusions and Future work

In this paper we discuss the scalability problem existing in recommender systems of OSNs and design a social recommender system named SRSH which is based on Hadoop. SRSH provides necessary recommendation services for users and can help the target user discover favorite information. Especially, we use the MapReduce programming model to parallelize core algorithms of SRSH to solve the scalability problem. Related experiments on Hadoop cluster platform show that SRSH has good scalability and is qualified to be the recommender

system in the realistic large-scale OSN. There still exist some drawbacks in SRSH. Therefore, in the future we are planning to continually improve every core recommendation modules of SRSH and let SRSH run on Hadoop platform more effectively.

Acknowledgements

This work was supported by the following projects: National High-Technology Research and Development Program ("863" Program) of China (Grant No. 2013AA01A212); National Natural Science Foundation of China (Grant No. 60970044, 61272067, 61370178); National Science and Technology Support Program of China (Grant No. 2012BAH27F05); Natural Science Foundation of Guangdong Province of China (Grant No. S2012030006242); Science and Technology Support Program of Guangdong Province of China (Grant No. 2012A080104019, 2011B080100031); Foundation for Distinguished Young Talents in Higher Education of Guangdong, China (Grant No. 2012LYM_0077).

References

[1] J. M. He and W. W. Chu, "A social network-based Recommender System (SNRS)", Data Mining for Social Network Data Annals of Information Systems, vol. 12, **(2010)**, pp. 47-74.
[2] E. Davoodi, M. Afsharchi and K. Kianmehr, "A social network-based approach to expert recommendation System", Lecture Notes in Computer Science, vol. 7208, **(2012)**, pp. 91-102.
[3] M. S. Pera, N. Condie and Y. K. Ng, "Personalized book recommendations created by using social media data", In Proceedings of the 2010 international conference on Web information systems engineering, Hong Kong, China, **(2010)** December, pp. 390-403.
[4] J. Golbeck and J. Hendler, "FilmTrust: Movie recommendations using trust in web-based social networks", In Proceedings of 3rd IEEE Consumer Communications and Networking Conference, Las Vegas, NV, USA, **(2006)** January, pp. 282-286.
[5] J. Lin. "Brute force and indexed approaches to pairwise document similarity comparisons with MapReduce", In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, Boston, USA, **(2009)** July, pp. 155-162.
[6] T. Elsayed, J. Lin and D. W. Oard, "Pairwise document similarity in large collections with MapReduce", In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies, Columbus, Ohio, USA, **(2008)** June, pp. 265-268.
[7] K. Shuang, Y. Yang, B. Cai and Z Xiang, "X-RIME: Hadoop-based large-scale social network analysis", In proceedings of 3rd IEEE International Conference on Broadband Network and Multimedia Technology, Beijing, China, **(2010)** October, pp. 901-906.
[8] G. J. Liu, M. Zhang and F. Yan, "Large-Scale social network analysis based on MapReduce", In Proceedings of 2010 International Conference on Computational Aspects of Social Networks, Taiyuan, China, **(2010)** September, pp. 487-490.
[9] P. Sharma, U. Khurana, B. Shneiderman, M. Scharrenbroich and J. Locke, "Speeding up network layout and centrality measures for social computing goals", Lecture Notes in Computer Science, vol. 6589, **(2011)**, pp. 244-251.
[10] J. Tang, J. M. Sun, C. Wang and Z. Yang, "Social influence analysis in large-scale networks", In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, **(2009)**, pp. 807-816.
[11] J. Travers and S. Milgram, "An Experimental Study of the Small World Problem", Sociometry, vol. 32, no. 4, **(1969)**, pp. 425-443.
[12] W. Frakes and R Baeza-Yates, "Information Retrieval: Data structures and Algorithms", Prentice-Hall, New Jersey, **(1992)**.
[13] L. Tang and H. Liu, "Graph mining applications to social network analysis", Managing and Mining Graph Data Advances in Database Systems, vol. 40, **(2010)**, pp. 487-513.
[14] J. Liu, W. Q. Wang, Z. Y. Chen, X. Z. Du and Q. Qi, "A novel user-based collaborative filtering method by inferring tag ratings", ACM SIGAPP Applied Computing Review, vol. 12, no.14, **(2012)**, pp. 48-57.
[15] A. Bellogin and J. Parapar, "Using graph partitioning techniques for neighbor selection in user-based collaborative filtering", In Proceedings of the 6th ACM conference on Recommender systems, Dublin, Irland, **(2012)** September, pp. 213-216.

[16] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, "Item-based collaborative filtering recommendation algorithms", In Proceedings of the 10th international conference on World Wide Web, Hong Kong, China, **(2001)** May, pp. 285-295.
[17] B. Rostami, P. Cremonesi and F. Malucelli, "A graph optimization approach to Item-Based collaborative filtering", Recent Advances in Computational Optimization Studies in Computational Intelligence, vol. 470, **(2010)**, pp.15-30.
[18] Flixster dataset, http://www.cs.sfu.ca/~sja25/personal/datasets/.
[19] Flixster, http://www.flixster.com/.

Authors

Chaobo He, he is an associate professor in ZhongKai University of Agriculture and Engineering. He is also a Ph.D. student at School of Computer, South China Normal University. His main research areas are data mining and social computing.

Yong Tang, he is a professor and Ph. D. supervisor in South China Normal University. His main research interests include big data, cloud computing, collaborative computing and academic information service.

Zhenxiong Yang, he is a graduate student in South China Normal University. His main research interests are social network analysis and mining.

Kai Zheng, he is a senior experimentalist in South China Normal University. His main research interests are data integration and E-Campus application.

Guohua Chen, he is a post doctor in South China Normal University. His main research interests are social network analysis and mining.