

Recommendation via Query Centered Random Walk on K-partite Graph

Haibin Cheng, Pang-Ning Tan, Jon Sticklen, William F. Punch
Michigan State University
East Lansing, MI 48824
{chenghai,ptan,sticklen,punch}@cse.msu.edu

Abstract

This paper presents a recommendation algorithm that performs a query dependent random walk on a k -partite graph constructed from the various features relevant to the recommendation task. Given the massive size of a k -partite graph, executing the query centered random walk in an on-line fashion is computationally infeasible. To overcome this challenge, we propose to apply multi-way clustering on the k -partite graph to reduce the dimensionality of the problem. Random walk is then performed on the subgraph induced by the clusters. Experimental results on three real data sets demonstrate both the effectiveness and efficiency of the proposed algorithm.

1 Introduction

Recommender systems are tools that automatically filter a large set of items (movies, books, scientific papers, music, jokes, etc) in order to identify those that are most relevant to user's interest. There are two common strategies adopted for making recommendations—content-based filtering and collaborative filtering. Content-based filtering utilizes the features of an item to determine its relevance whereas collaborative filtering makes its recommendation based on ratings provided by other users with similar interests.

Recently there have been considerable interests in applying graph-based methods for generating recommendations [8]. These methods represent the users and items as nodes of a bipartite graph and employ a Markov chain random walk algorithm to rank the nodes based on their authoritativeness. Random walk approaches are advantageous because they consider the relationships between users and items from a global perspective, as opposed to the local pairwise similarity methods used by standard content-based and collaborative filtering approaches.

Besides information about users and items, other domain features are often available to improve the recommendations. For example, when recommending scientific

papers, the additional features include author names, keywords, publication venue, and reference citations of the papers. A natural way for incorporating these multivariate features is by using a k -partite graph. Applying random walk on a k -partite graph however is extremely costly, not only due to the massive graph size but also because the random walk must be guided by the features relevant to each user.

In this paper, we propose a graph-based recommendation algorithm that performs a query dependent random walk on a k -partite graph. We empirically demonstrate the benefits of using the proposed algorithm compared to content-based filtering, collaborative filtering, and random walk on bipartite graph approaches. To improve its efficiency, we apply multi-way clustering to reduce the dimensionality of the problem. Our clustering approach is based on the idea of decomposing the adjacency matrices of the k -partite graph using the non-negative matrix factorization approach [1]. A query centered random walk is then performed on the subgraph induced by the clusters. Experiments performed on three real-world data sets demonstrate the efficiency and effectiveness of our proposed algorithms.

2 Preliminaries

A k -partite graph is a graph whose graph vertices can be partitioned into k disjoint sets so that no two vertices within the same set are adjacent. It provides a natural way for modeling heterogeneous data sets. For example, in scientific paper recommendation, the graph represents the various types of features of a scientific paper such as terms in titles and abstracts, authors, publication venues, and reference citations. In movie recommendation, the graph contains information such as movie titles, movie genres, MPAA ratings, actors, directors, and user preference ratings.

Let $G = \{V_1, V_2, \dots, V_k, E\}$ be a k -partite graph, where each V_i corresponds to a set of nodes of the same type and E is the set of edges connecting nodes of different types. For brevity, the discussion in this paper focuses on tripartite graphs, although the formulation can be extended beyond $k = 3$.

A tripartite graph is a graph whose graph vertices can be partitioned into 3 disjoint sets so that no two vertices within the same set are adjacent. For example, in a movie recommendation domain, the sets of nodes may correspond to users, movies, and genre, while in scientific paper recommendation, the sets of nodes may correspond to terms, documents, and authors. The relationships encoded by the edges between two sets of nodes can be summarized into a pair of adjacency matrices, P and Q . P denote the adjacency matrix for the term-document edges, while Q denote the corresponding matrix for document-author edges. The edges can also be assigned weights; e.g., to represent the frequency of a term in a document or the order of authors listed in a document. A $|V| \times |V|$ adjacency matrix M for the tripartite graph, $G = \{V^t, V^d, V^a, E\}$ is constructed as follows:

$$M = \begin{bmatrix} 0 & P & 0 \\ P^T & 0 & Q \\ 0 & Q^T & 0 \end{bmatrix} \quad (1)$$

where

$$P_{ij} = \begin{cases} e_{ij}, & e_{ij} \in E, v_i \in V^t, v_j \in V^d; \\ 0, & \text{otherwise.} \end{cases}$$

$$Q_{ij} = \begin{cases} e_{ij}, & e_{ij} \in E, v_i \in V^d, v_j \in V^a; \\ 0, & \text{otherwise.} \end{cases}$$

where $|V| = |V^t| + |V^d| + |V^a|$.

3 Query Centered Random Walk

A Markov chain random walk model can be defined for the k-partite graph by transforming its adjacency matrix M into a transition probability matrix S using column normalization:

$$S_{ij} = \frac{M_{ij}}{\sum_i M_{ij}} = \begin{bmatrix} 0 & P_1 & 0 \\ P_2 & 0 & Q_1 \\ 0 & Q_2 & 0 \end{bmatrix} \quad (2)$$

where:

$$(P_1)_{ij} = \frac{P_{ij}}{\sum_i P_{ij} + \sum_k Q_{jk}}$$

$$(P_2)_{ij} = \frac{P_{ji}}{\sum_i P_{ji}}$$

$$(Q_1)_{ij} = \frac{Q_{ij}}{\sum_i Q_{ij}}$$

$$(Q_2)_{ij} = \frac{Q_{ji}}{\sum_i P_{ij} + \sum_k Q_{jk}}$$

Each node of the graph is assumed to be a given state of a stochastic process. The transition matrix S would govern the probability that the process goes from one state to another.

In a query centered random walk, the transition also depends on an input profile vector q . For recommender systems, q corresponds to a profile vector capturing the user preferences. For example, in scientific paper recommendation, q indicates the terms describing the research interest of a user or the initial list of papers and authors preferred by the user. Such prior information is incorporated into the un-normalized profile vector \hat{q} as follows:

$$\hat{q} = \begin{bmatrix} \hat{q}^t \\ \hat{q}^d \\ \hat{q}^a \end{bmatrix}$$

whose elements are:

$$\hat{q}_i = \begin{cases} 1, & \text{if } v_i \text{ is a preferred node in the k-partite graph;} \\ 0, & \text{otherwise.} \end{cases}$$

The random walk model begins from a node, v_0 , chosen according to an initial probability vector r_0 . Next, with probability α , the random surfer may visit one of the preferred nodes in q , or with probability $1 - \alpha$, transits to one of its neighboring nodes in the graph. Such a model can be summarized by the following matrix equation:

$$r = (1 - \alpha)Sr + \alpha q, \quad (3)$$

where q is the normalized profile vector. The random walk model allows us to compute the probability that the random surfer is at a particular node after k steps. By iteratively applying Equation 3 until convergence, the stationary distribution for r can be used to rank the nodes (terms, documents, and authors) based on their importance. Documents that are ranked highly can be recommended to the user if they do not belong to the initial profile vector q .

For k-partite graphs, Equation (3) can be further simplified as follows:

$$r^t = (1 - \alpha)P_1 r^d + \alpha q^t$$

$$r^d = (1 - \alpha)(P_2 r^t + Q_1 r^a) + \alpha q^d$$

$$r^a = (1 - \alpha)Q_2 r^d + \alpha q^a \quad (4)$$

The query centered random walk algorithm for k-partite graph is referred to as **QRank** in this paper.

The QRank algorithm has some desirable properties. The Markov chain model computes the relevance score of each document from a global perspective, with respect to all the documents, terms, and authors in the graph. In contrast, standard approaches consider only the local properties (e.g., by comparing the similarity of a document to the user preference vector).

One caveat is that the probability vector r must be computed for each user. Although there are efficient algorithms available for doing sparse matrix computations, the algorithm is still computationally expensive especially if the size

of the graph is very large (e.g., when the data contains thousands of authors, millions of documents, and hundreds of thousands of terms). One way to alleviate the problem is to perform the computation offline, but this would prevent the users from updating their profile in a real-time fashion. In the following section, we will introduce a more efficient implementation by partitioning the nodes of the k-partite graph offline and then applying the QRank algorithm on a considerably smaller subgraph.

4 Improving Efficiency via Multi-way k-partite Graph Clustering

When applying the random walk algorithm described in the previous section to a k-partite graph, we observe that the recommended nodes are generally centered around the neighborhood of the queried nodes in the profile vector q^t . For example, the documents and authors that have high scores tend to cluster around terms that appear in the profile vector. This motivates us to develop a multi-way clustering approach to reduce the dimensionality of the random walk problem and thus, improve its efficiency so that it is applicable to a real-time setting.

4.1 Multi-way Clustering on k-partite Graph

Consider a tripartite graph built from a corpus of literature publications. Assume the graph contains m term nodes, n document nodes and h author nodes. Let P and Q be the corresponding term-document and document-author adjacency matrices.

The objective of multi-way clustering is to simultaneously group the term nodes into k_1 term clusters, document nodes into k_2 document clusters, and author nodes into k_3 author clusters. Figure 1 illustrates the basic idea behind this method. H1, H2, and H3 correspond to clusters of terms, H4 and H5 are clusters of documents, while H6 and H7 are clusters of authors. Adjacency matrices are created between every two sets of nodes. Let $U \in \mathbb{R}^{m \times k_1}$ be the adjacency matrix for edges connecting the m term nodes to the k_1 term clusters, $F \in \mathbb{R}^{k_1 \times k_2}$ be the adjacency matrix for the edges connecting the k_1 term cluster to the k_2 document clusters, and $D \in \mathbb{R}^{n \times k_2}$ be the adjacency matrix for the edges connecting the n document nodes to k_2 the document clusters. Furthermore, let $S \in \mathbb{R}^{k_2 \times k_3}$ be the adjacency matrix for edges connecting the k_2 document clusters to the k_3 author clusters while $R \in \mathbb{R}^{h \times k_3}$ be the adjacency matrix for the edges connecting the h author nodes to the k_3 author clusters.

The objective function of our multi-way clustering is to learn the matrices U , F , D , S and R in such a way that

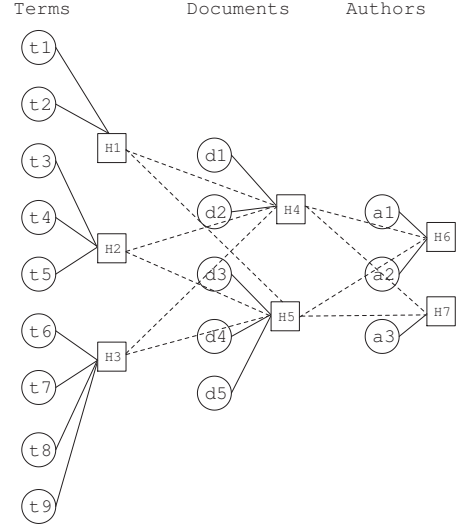


Figure 1. Multi-way clustering on k-partite graph. The nodes labeled as H1, H2, \dots , H7 correspond to clusters of terms, documents, and authors.

minimizes the sum squared errors between the original matrices (P and Q) with the factorized matrices (UFD^T and DSR^T):

$$\begin{aligned} \min \quad & \|P - UFD^T\|^2 + \|Q - DSR^T\|^2 \\ \text{s.t.} \quad & U^T U = I, D^T D = I, R^T R = I \\ & U, F, D, S, R \geq 0 \end{aligned} \quad (5)$$

Note that the preceding cluster formulation is similar to the Bregman divergence k-partite graph clustering problem posed by Long et al. in [2]. However, unlike their work, which uses a variation of the k-means algorithm to solve the clustering problem, we propose to optimize our objective function using non-negative matrix factorization (NMF) approach. In previous studies by Ding et al. [1], it has been shown that NMF generally performs better than k-means in terms of its flexibility, cluster quality, and ability to provide good matrix approximations.

Note that the tri-factorized matrices U , D , and R provide information about the cluster memberships of terms, documents, and authors. To solve the constrained optimization problem, we introduce three types of Lagrangian multipliers, $\lambda_1 \in \mathbb{R}^{k_1 \times k_1}$, $\lambda_2 \in \mathbb{R}^{k_2 \times k_2}$ and $\lambda_3 \in \mathbb{R}^{k_3 \times k_3}$. The corresponding Lagrangian function for the optimization problem is:

$$\begin{aligned} L = \quad & \|P - UFD^T\|^2 + \|Q - DSR^T\|^2 \\ & + Tr[\lambda_1(U^T U - I)] + Tr[\lambda_2(D^T D - I)] \\ & + Tr[\lambda_3(R^T R - I)] \end{aligned} \quad (6)$$

The preceding objective function is an extension to the bi-orthogonal tri-factorization problem described by Ding et al. [1]. Instead of factorizing one matrix, our objective function requires the factorization of two matrices, P and Q . The problem is further complicated by the fact that the matrix D is a common factor in the decomposition of P and Q .

For matrices U and F , since they participate only in the decomposition of matrix P (i.e., the first term of the objective function), we can use the iterative update algorithm to learn the values of the matrices. The update rules for U and F are:

$$U_{ik} \leftarrow U_{ik} \sqrt{\frac{(PDF^T)_{ik}}{(UU^T PDF^T)_{ik}}} \quad (7)$$

$$F_{ik} \leftarrow F_{ik} \sqrt{\frac{(U^T PD)_{ik}}{(U^T U F D^T D)_{ik}}} \quad (8)$$

Similarly, since S and R participates only in the decomposition of matrix Q (i.e., the second term of the objective function), we can use the iterative update algorithm to learn their values:

$$S_{ik} \leftarrow S_{ik} \sqrt{\frac{(D^T QR)_{ik}}{(D^T D S R^T R)_{ik}}} \quad (9)$$

$$R_{ik} \leftarrow R_{ik} \sqrt{\frac{(Q^T DS)_{ik}}{(R R^T Q^T DS)_{ik}}} \quad (10)$$

The correctness and proof of convergence of these update formulas are similar to those provided in [1] and thus are ignored in this paper.

For the common factor D , which participates in the decomposition of both P and Q , we need to determine its update formula separately. From Equation (6), the Lagrangian function can be expressed as following:

$$\begin{aligned} L = & Tr(P^T P - 2P^T U F D^T + D F^T U^T U F D^T) \\ & + Tr(Q Q^T - 2Q R S^T D^T + D S R^T R S^T D^T) \\ & + Tr[\lambda_1(U^T U - I)] + Tr[\lambda_2(D^T D - I)] \\ & + Tr[\lambda_3(R^T R - I)] \end{aligned} \quad (11)$$

Taking the gradient of L with respect to D , we obtain:

$$\begin{aligned} \frac{\partial L}{\partial D} = & -2P^T U F + 2D F^T U^T U F - 2Q R S^T \\ & + 2D S R^T R S^T + 2D \lambda_2 \end{aligned} \quad (12)$$

with the corresponding KKT complementarity condition:

$$\begin{aligned} (-2P^T U F + 2D F^T U^T U F - 2Q R S^T \\ + 2D S R^T R S^T + 2D \lambda_2)_{ik} D_{ik} = 0 \end{aligned} \quad (13)$$

to ensure that the local minimum holds.

Using the following update rule,

$$D_{ik} \leftarrow D_{ik} \sqrt{\frac{(P^T U F + Q R S^T)_{ik}}{[D(F^T U^T U F + S R^T R S^T + \lambda_2)]_{ik}}} \quad (14)$$

it can be shown that the solution converges to a fix point that satisfies the following condition:

$$\begin{aligned} (-2P^T U F + 2D F^T U^T U F - 2Q R S^T \\ + 2D S R^T R S^T + 2D \lambda_2)_{ik} D_{ik}^2 = 0 \end{aligned}$$

which has similar form as the condition stated in (13). The detailed proof is provided in the technical report ??.

4.2 Query-Centered Random Walk on Clustered K-partite Graph

The derived clusters not only provide a way to group together related nodes, it also helps to reduce the computational complexity of performing a query centered random walk on large k-partite graphs. For example, given a user preference vector q , we first identify all the term clusters associated with the terms in q based on the cluster membership matrix U . Once we have identified the sub-matrix U^{sub} corresponding to the terms in q , we may find the corresponding document clusters (and its associated sub-matrix D^{sub}) by examining the matrix F , which relates the term clusters to document clusters. Similarly, we can obtain the corresponding author clusters (R^{sub}) via the matrix S , which relates the document clusters to author clusters.

Once the sub-matrices U^{sub} , D^{sub} and R^{sub} are found, a subgraph of the k-partite graph is constructed. Because the size of the subgraph is considerably smaller than the original graph, a query centered random walk can be efficiently performed in a real time fashion to identify the relevant documents. The detailed steps of our algorithm, which is known **QCRank**, is presented in Figure ??,

Clearly there is a tradeoff between computational efficiency and precision of the ranking results depending on the size of the constructed subgraph. The larger the subgraph is, the better the precision is, at the expense of increasing computational time. As will be shown in our experimental results section, it is possible to find reasonable thresholds for identifying the relevant clusters such that a fast random walk can be performed using QCRank without sacrificing the high precision of QRrank.

5 Experimental Evaluation

This section describes the experiments performed to demonstrate the benefits of using our proposed query centered random walk recommendation algorithm (**QRrank**) along with its online (**QCRank**). All of our experiments were performed on a 3.6 GHz Windows XP machine with 1 GB RAM.

QCRank Algorithm:
Offline Partition by Non-negative Matrix Factoring
Input: Tripartite graph with term-document matrix P and document author matrix Q
Output: Term cluster membership matrix U , document cluster membership matrix D and author cluster membership matrix R , term-document cluster relevance matrix F and document-author relevance matrix S
Method:
1. Updating U, F, D, S, R by the rules in (7), (8), (14), (9), (10) until converge.
2. Store the matrices in the disk.
Online Random Walk Recommendation
Input: Query q^t and Matrices U, F, D, S, R
Output: List of ranked documents r^d , ranked terms r^t and r^a
Method:
1. Find the cluster subset U^{sub}, D^{sub} and R^{sub} by searching the matrices U, F, D, S, R .
2. Build subgraph with term-document matrix P^{sub} and document-author matrix Q^{sub} .
3. Run the random walk algorithm in (5)

5.1 Experimental Setup

We conducted our experiments using two data sets: OHSUMED [6] and MOVIELENS [7]. The OHSUMED test collection is a benchmark data set used to evaluate the performance of recommender systems. We created a sparse term-document-author tripartite graph from the data. The MOVIELENS data set consists of 100,000 movie ratings provided by 943 users on 1682 movies. A user-movie-category tripartite graph is then constructed from the data.

We consider both content-based and collaborative filtering algorithms as the baseline methods of our algorithm. The content-based methods include cosine similarity [3], Okapi [4] and K-L Divergence [5]. We tested the OHSUMED data with all three baseline methods. For the MOVIELENS data, since user ratings are available, we compared our proposed methods against standard collaborative filtering approach.

5.2 Experimental Results

This section compares the performance of our algorithm against other baseline methods. QRank is our full random walk algorithm on the k-partite graph while QCRank is its online version, which performs random walk on a subgraph induced by the clusters in the input profile vector. To illustrate the benefits of performing random walk on tripartite graphs, we also compare our algorithms against QRank-2, which performs random walk on a bipartite graph constructed from the term-document matrix.

	OHSU1	OHSU2	OHSU3	OHSU4
cosine	0.305	0.286	0.314	0.268
okapi	0.292	0.291	0.298	0.281
K-L	0.291	0.303	0.315	0.278
QRank-2	0.295	0.301	0.318	0.297
QRank	0.304	0.309	0.322	0.301
QCRank	0.302	0.302	0.315	0.297

Table 1. Comparison of average precision for four subsets of the OHSUMED data.

	movieLen
CF	0.1757
QRank-2	0.1764
QRank	0.1781
QCRank	0.1758

Table 2. Comparison of rank correlation coefficients for MOVIELENS data.

Table 1 shows the experimental results for the OHSUMED data. For each of the four subsets of data, we repeated the experiment 10 times, each time using only 100 queries. The results reported in the table correspond to the average precision for ten runs. The results clearly suggest that graph-based methods generally outperform all three content-based filtering approaches. Furthermore, QRank is generally better than QRank-2, thus illustrating the advantages of incorporating authorship information and modeling the problem using tripartite graphs instead of bipartite graphs. Finally, observe that QCRank, which is an efficient online approach, is comparable and sometimes outperforms the QRank-2 approach.

Finally, for the MOVIELENS data, we compare the performance of our algorithms against the collaborative filtering approach. The results are reported in Table 2 after performing 4-fold cross validation. Since the ratings provided by users are limited to integers in the range between 1 and 5, many movies would have identical ratings. So, instead of comparing their absolute rating scores, we consider the relative ordering between all pairs of movies in the test set (whether movie 1 is ranked higher, lower, or the same as movie 2). We then employ Kendall’s τ rank correlation coefficient as our evaluation measure [?]. Again, the results in Table 2 suggest that our QRank algorithm outperforms both collaborative filtering and the bipartite graph-based (QRank-2) approaches.

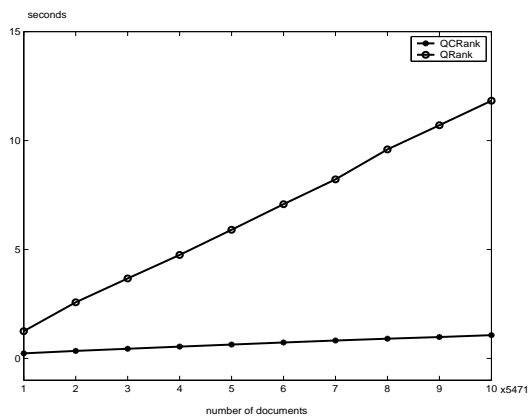


Figure 2. Runtime comparison between the full random walk and its online algorithm.

5.3 Efficiency of Proposed Algorithms

To investigate the relative efficiencies of the proposed algorithms, we created ten OHSUMED data sets with varying number of documents, increasing from 5471 to 54710 as shown in the x-axis of Figure Figure 2. The y-axis in Figure 2 shows the runtime for each data set using the QCRank and QRank algorithms. It is obvious that as more nodes are added into the graph, the runtime for both random walk algorithms increases due to the increasing size of the adjacency matrices they have to process. Nevertheless, the QCRank algorithm, which uses multi-way clustering to identify an induced subgraph for which the random walk algorithm is applied, is considerably more efficient and scalable than the full-blown QRank algorithm.

6 Conclusions

Recommender systems often benefit from utilizing a variety of features relevant to the recommendation task. A k-partite graph provides a natural way for representing such heterogeneous features. This paper presents a recommendation algorithm for k-partite graphs using the query centered random walk approach. Experiments performed on three real-world data sets demonstrate the effectiveness of our algorithm in comparison to other existing methods such as content-based filtering and collaborative filtering.

However, the massive size of k-partite graphs often makes it infeasible to apply random walk methods on large data sets. To overcome this problem, we propose to reduce the dimensionality of the problem using multi-way clustering on k-partite graph. A query centered random walk can then be performed on the subgraph induced by the clusters. Our experiments confirmed the improved speedup achieved

using this approach. Our proposed approach also identifies additional terms that can be augmented into the profile vector to further enhance the performance of the algorithm.

References

- [1] Chris Ding, Tao Li, Wei Peng, Haesun Park. Orthogonal Nonnegative Matrix Tri-factorizations for Clustering, *Proc Int'l Conf. on Knowledge Discovery and Data Mining*, August, 2006.
- [2] Bo Long, Xiaoyun Wu, Zhongfei (Mark) Zhang, and Philip S. Yu. Unsupervised Learning on K-partite Graphs, *Proc. ACM International Conference on Knowledge Discovery and Data Mining*, ACM Press, Philadelphia, PA, USA, August, 2006.
- [3] Baeza-Yates, R. Ribeiro-Neta, B. *Modern information retrieval*. New York : ACM Press, 1999.
- [4] S.E. Robertson, S. Walker, M.M. Beaulieu, M. and Gatford, A. Payne. Okapi at TREC-4, *In The Fourth Text Retrieval Conference (TREC-4)*, 1993.
- [5] Zhai, C. and Lafferty, J. Model-based feedback in the KL-divergence retrieval model, *In Tenth International Conference on Information and Knowledge Management*, pp. 403-410, 2001.
- [6] Hersh WR, Buckley C, Leone TJ, Hickam DH, OHSUMED: An interactive retrieval evaluation and new large test collection for research *Proceedings of the 17th Annual ACM SIGIR Conference*, pp. 192-201, 1994.
- [7] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. GroupLens. An open architecture for collaborative filtering of netnews. *In Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, Chapel Hill, NC, 1994.
- [8] M. Gori and A. Pucci, ItemRank, A Random-Walk Based Scoring Algorithm for Recommender Engines, *20th International Joint Conference on Artificial Intelligence* Hyderabad, India 2007.
- [9] R. Bekkerman, R. El-Yaniv and A. McCallum. Multi-Way Distributional Clustering via Pairwise Interactions. *In Proceedings of the 22nd International Conference on Machine Learning*, pp. 41-48, 2005.
- [10] Haibin Cheng, Pang-Ning Tan, Jon Sticklen, William F. Punch. Recommendation via Query Centered Random Walk on K-partite Graph, *Technical Report*, MSU-CSE-07-190.