RankCompete: Simultaneous Ranking and Clustering of Information Networks

Liangliang Cao[a], Xin Jin[b], Zhijun Yin[b], Andrey Del Pozo[a], Jiebo Luo[c], Jiawei Han[b], Thomas S. Huang[a]

[a]*Beckman Institute and Coordinated Science Lab,University of Illinois at Urbana-Champaign*
[b]*Dept. Computer Science,University of Illinois at Urbana-Champaign*
[c]*Kodak Research Laboratories, Rochester, USA*

Abstract

Random walk was first introduced by Karl Pearson in 1905 and has inspired many research works in different fields. In recent years, random walk has been adopted in information network research, for example, ranking and similarity estimation. In this paper, we introduce a new model called RankCompte, which allows multiple random walkers in the same network (existing work mostly focus on random walks of a single category). By introducing the "competition" concept into the random walk framework, our method can fulfill both clustering and ranking tasks simultaneously and thus provide an effective analysis tool for networks. Compared with the traditional network ranking approaches, our new method focuses more on the structure of specialized clusters. Compared with the traditional graph clustering approaches, our new method provides a faster and more intuitive way to group the network nodes. We validate our approach on both bibliography networks and visual information networks, and the results show that our approach can obtain 100% perfect clustering results in clustering the DBLP 20 conferences and outperform the state-of-the-art on the Cora dataset. Furthermore, we show that our method can be effectively used to summarize personal photo collections.

Keywords: Clustering, Ranking, Information Network

1. Introduction

In recent years, there have been an increasing amount of interests in studying information networks and social networks, including online community, transportation network, research collaboration network, social media network, and so on. Two kinds of information network are used in this paper: First is the

conference-author network, where a link between a conference and an author denotes that the author has published a paper in the conference. Second is the visual information network, where a link between images denotes that the two images share similar visual content (e.g., local image patches). To understand such complicate information networks, we consider the following problems need to be answered.

1. *Clustering*: Among the multiple types of nodes in an information network, we often need to group subject nodes into clusters. For example, in the conference-author network, a conference node should be associated with a unique clustering label corresponding to a research area (e.g., data mining, machine learning or database).

2. *Ranking*: In information networks, it is often interesting to compare the importance of nodes. For example, given an author-conference network, can a data mining algorithm judge whether VLDB is more important than ICDE? From a collection of personal photos, is there a way to automatically find representative photos and throw away unimportant ones if necessary? A ranking algorithm can help solve this problem in information networks.

In this paper,these two problems will be address in an unified viewpoint based on random walk models. A random walk is a mathematical formalization of a trajectory that consists of taking successive random steps. It was first introduced by Karl Pearson in 1905. Since then, it has inspired many studies related to electric circuit, harmonic field, diffusion model, and so on. It has become a fundamental topic in discussions of Markov processes, in which a random walk is often characterized by a transition probability matrix from one element to another. A random walker can jump from element to element, which can be represented by a Markov chain.

The most well-known algorithm related to random walk is PageRank [1], which leads to a revolution in web searching engines. The main idea of PageRank is to view a link from one page to another as an endorsement of the landing page. The more links that point to a page, the more likely it is important. The importance of each page is modeled by a ranking score vector $\mathbf{p}$, which can be obtained effectively by iteratively updating $\mathbf{p} \leftarrow \mathbf{T}\mathbf{p}$. Similarly, Kleinberg [2] proposed to introduce two scores for each node, hub and authority, and developed the HITS algorithm that in spirit resembles PageRank. Following these studies, efforts have been reported that introduce expert knowledge to improve searching results [3], and build topic-sensitive PageRank [4]. These random walk based algorithms have dramatically extended our ability to organize web documents.

The essence of random walk has also been applied to semi-supervised learn-

ing. In [5], Zhou *et al.* proposed to combine a fixed vector $\mathbf{y}$ to estimate the ranking scores by $\mathbf{p} \leftarrow \alpha \mathbf{T}\mathbf{p} + (1-\alpha)\mathbf{y}$, where $\mathbf{y}$ embeds information of partial labels, and α is a weighting coefficient.

Existing studies mostly focus on random walk of a single category. In this paper, we introduce the "competition" concept into the random walk framework by allowing multiple walkers in the same network. Our new model is named *RankCompte* model, which provides an effective tool for analyzing not only traditional graphs but also more complicated information networks.

The RankCompete model proposed in this paper can fulfill the two tasks simultaneously. By allowing multiple random walks to compete iteratively, our RankCompete model simultaneously clusters and ranks the nodes in information networks. On one hand, the random walks can be updated very quickly and the clustering results can be easily inferred from the competition procedure. On the other hand, the RankCompete model estimates the ranking scores within each cluster, which is more reasonable than ranking the nodes across different clusters (a la comparing "oranges" and "apples"). For example, in the author-conference network, it does not make much sense to rank authors from different areas together, e.g., physics and computer science. It is more meaningful to rank authors in the same research areas.

1.1. Related Work

Since our RankCompete models are designed to perform simultaneously ranking and clustering for information networks, it is necessary to compare our model with some recent clustering and ranking algorithms. In this section, we first go over the related work and then summarize the differences with our method in Table 1.

Our approach is motivated by the success of PageRank algorithm, which not only upholds the efficiency guarantees but also enhances it with the ability to distinguish network clustering. Later Hilltop [3] and Topic-Sensitive Page Rank (TSPR) [4] have been proposed to improve the accuracy of PageRank. Our approach is different from these methods since we are pursuing ranking functions in local neighborhood, while the previous works consider global ranking.

Spectral clustering algorithms [6] [7] are proposed to cluster data using eigenvector of matrix derived from graphs. All these works assume that the similarity between two nodes can be reliably computed. On the other hand, the similarity between heterogeneous nodes in the information network is usually not well defined. Moreover, spectral clustering usually requires computing k eigenvectors, which can be time consuming.

Table 1: Comparing our RankCompete model with other graph analysis algorithms.

Method	time complexity	clustering number	ranking scores
Normalized Cuts	$O(N^{3/2})$	fixed k	$\times$
Spectral Clustering	$O(N^3)$	fixed k	$\times$
RankClus	$O(m(c_1N + c_2N))$	fixed k	$\surd$
RankCompete	$O(c_1N)$	either fixed k or hierarchical	$\surd$

The most similar work to this paper is RankClus by Sun *et al.* in [8]. They first initialize k clusters and then represent each author with a k-dimensional vector with each dimension accounting for the ranking score with respect to a cluster. Then an EM-like algorithm based on the new feature vectors is performed to update the clusters. Unlike [8] which requires initialized clusters and uses a mixture model to represent each document, our algorithm starts with a few seed nodes, and ranking and clustering are updated simultaneously as two groups of ranking functions compete with each other in the information network. In each iteration of RankClus, the algorithm needs to update the ranking scores until they converge and perform an EM algorithm. Suppose it takes m iterations for RankClus to converge, the complexity of RankClus is $O(m(c_1N + c_2N))$, where c_1 and c_2 are the number of ranking score updates and EM estimations in each iteration, respectively. In contrast, our algorithm has a complexity of only $O(c_1N)$.

2. Problem Formulation

Figure 1 illustrates an abstract model for information networks. The rectangular boxes denote the subject nodes in the network, while the other boxes correspond to the attributes of the subjects. In this paper, the subject nodes are named as *s-nodes*, while the attribute nodes are called *a-nodes*. For example, in a conference-author network, the subject nodes are different conferences, with attributes like authors, key words, and paper titles. In a visual information network, subject nodes correspond to images, with attributes such as local patches, tags, metadata information, and so on. We introduce multiple random walks in the figure, and the competition of these random walks will help us rank and cluster the subjects. Next we will review the concepts for information network and discuss our model.

A *Random Walk* (RW) is the trajectory of a walker that takes random steps on subject nodes. In our approach, we suppose there are K walkers $RW_1, RW_2, \ldots, RW_K$.

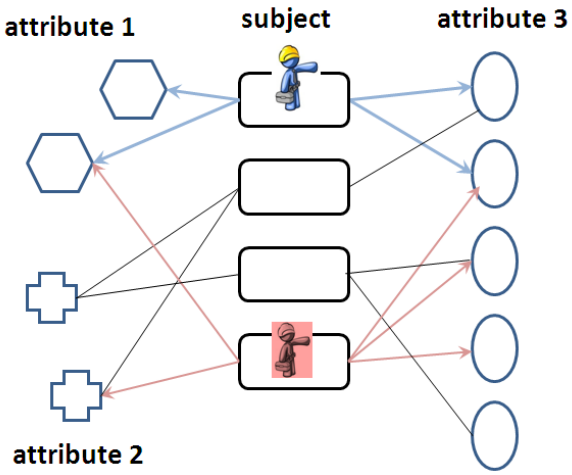

Figure 1: Abstract model for information network and RankCompete.

Given a random walk RW_k, the *random walk score* $p_k(u)$ for a node u is defined as the probability of visiting node u. The random walk scores should satisfy the constraint

$$\sum_{u=1}^{N} p_k(u) = 1 \tag{1}$$

where N is the number of s-nodes. The *state* of a random walker is determined by the probability of visiting each node in the graph. Given N nodes, a state is modeled by a vector $\mathbf{p}_k = [p_k(1), \cdots, p_k(N)]$.

The *transition matrix* $\mathbf{T}$ is the most important concept for random walks, which defines the probability of a random walk to travel from one node to another. Suppose the random walk score in the current state is $p^t(u)$, we can estimate $p^{t+1}(u)$ by $p^{t+1}(u) = \sum_v T(u,v)p^t(v)$, where $T(u,v)$ denotes the element at u row and v column in the transition matrix. Given the similarity $S(u,v)$ between two nodes u and v, we can write $T(u,v) = \frac{S(u,v)}{\sum_{v'} S(u,v')}$. In the matrix form, we have

$$\mathbf{T} = \mathbf{D}^{-1}\mathbf{S} \tag{2}$$

where $\mathbf{D}$ is a diagonal matrix with $D(u,u) = \sum_v S(u,v)$.

In an information network, there are three ways to define the similarity and transition matrix: (1) based on node features, (2) based on the network structures, or (3) linear weighting of multiple transition matrices over different attribute.

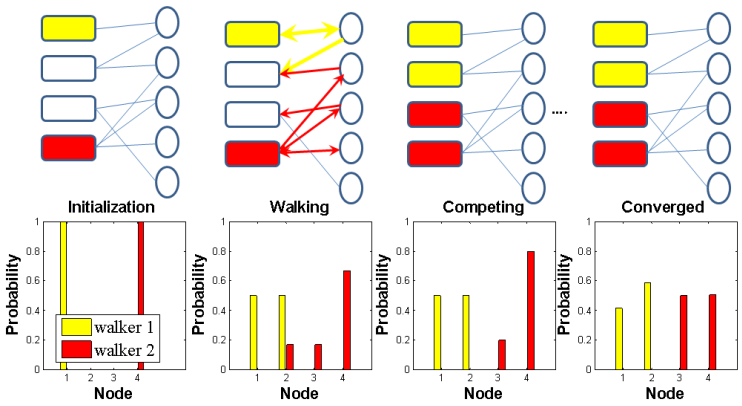

Figure 2: RankCompete on a toy information network. The yellow and red colors denote two different random walks. The top row shows the updating process in the network, and the bottom row displays the corresponding probability on the subject nodes in each step.

Our RankCompete employs multiple random walks $\{\mathbf{p}_k\}$, $1 \leq k \leq K$. For each node u, there are K random walk scores: $[p_1(u), \cdots, p_K(u)]$. The competing property requires that each node can host at most one random walk, in other words

$$p_k(u) = 0, \quad \text{if} \quad k \neq \arg\max_{k'} p_{k'}(u) \tag{3}$$

which means there is at most one non-zero random walk score for each node.

3. Our Models

3.1. RankCompete Model

Given a transition matrix $\mathbf{T}$, to cluster the s-nodes into K clusters, we introduce K random walkers with the random walk score vector $\{\mathbf{p}_k\}$, $1 \leq k \leq K$, which should satisfy the constraints in (1) and (3).

For an intuitive illustration, we employ a toy example with two random walkers. As shown in Figure 2, there are 4 subject nodes (rectangles) with 5 attribute nodes (circles). The two random walks $\mathbf{p}_1$ and $\mathbf{p}_2$ are denoted by the colors yellow and red color, respectively. In each iteration, each random walker will update the probability of visiting each node based on the transition matrix, and then they will compete to occupy the subject node. The cluster index will be determined by the color of subject nodes, and the probability in the second row shows the values of $\mathbf{p}_1$ and $\mathbf{p}_2$, corresponding to the ranking scores.

At the beginning, two different subjects are initialized as two categories (top and right-hand nodes in the first plot). In this case, the initialized $\mathbf{p}_1 = [1, 0, 0, 0]$, $\mathbf{p}_2 = [0, 0, 0, 1]$.

The RankCompete algorithm can be viewed as a two-step process. In the *walking* step, we update the random walk score vector $\mathbf{p}_k$

$$\overline{\mathbf{p}}_k = \mathbf{T}\mathbf{p}_k \tag{4}$$

Note that this iterative update works in a way similar to PageRank. The second column in Figure 2 shows $\overline{\mathbf{p}}_1$, $\overline{\mathbf{p}}_2$ changed after the first iteration.

In the *competing* step, the random walks $\overline{\mathbf{p}}_1, \overline{\mathbf{p}}_2, \ldots, \overline{\mathbf{p}}_K$ will compete on each node. At each subject node, the walker with the larger score will win the competition, while the other walker will fail to occupy that node and the corresponding score is set as zero. Mathematically, we update $\mathbf{p}_k$ by

$$p_k(u) = \begin{cases} \overline{p}_k(u), & \text{if } \overline{p}_k(u) = \max_{k'} \overline{p}_{k'}(u) \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

To guarantee that $\mathbf{p}_k$ satisfies the constraint of a random walk, we use a normalization process of

$$p_k(u) = p_k(u)/\rho_k \tag{6}$$

where $\rho_k = \sum_u p_k(u)$ is a normalization factor.

The third column of Figure 2 illustrates the competing process of random walks. The color of the box denotes the corresponding cluster associated with each subject.

RankCompete will iteratively update $\mathbf{p}_k$ based on Eqs. (4), (5), (6) until $\mathbf{p}_k$ no longer changes or the number of iterations exceeds a threshold (50 in our experiment). The fourth column of Figure 2 illustrates the converged RankCompete scores. We can see the converged results generate the ranking and clustering results simultaneously: The ranking scores are obtained from $\mathbf{p}_k$ directly, while the the cluster index for node u is computed as $k^* = \arg\max_k p_k(u)$.

Algorithm 1 summarizes the process of our RankCompete algorithm.

Next, we will discuss how the ranking scores of RankCompete are related to ranking. To make the presentation simple, we use two-class clustering for the discussion. Our results show that our RankCompete model leads to a PageRank score in a subgraph, Suppose the solution for RankCompete is $\mathbf{p}_1$ and $\mathbf{p}_2$ after convergence, we have

$$\overline{\mathbf{p}}_1 = \mathbf{T}\mathbf{p}_1, \quad \overline{\mathbf{p}}_2 = \mathbf{T}\mathbf{p}_2 \tag{7}$$

Algorithm 1 : RankCompete algorithm

1: **Input**: Number of clusters K.
2: **Initialize**: Select K s-nodes that are most separated as the starting node. Set $p_k(k) = 1$ for $1 \leq k \leq K$. Set all the other $\mathbf{p}$ to zero.
3: Set $iter = 1$
4: Update until convergence or $iter > 40$
5: *Walking step*: update $\overline{\mathbf{p}}_k$ based on (4).
6: *Competing step*: obtain $\mathbf{p}_k$ based on (5) and (6). and then update $\mathbf{T}_k$.
7: $iter = iter + 1$.
8: **Output**: $\mathbf{p} \in \mathbb{R}^{N \times K}$, the random walk scores and the cluster index.

and

$$\begin{array}{ll} p_1(u) = 0, & \text{if } \overline{p}_1(u) < \overline{p}_2(u), \\ p_2(u) = 0, & \text{otherwise.} \end{array} \tag{8}$$

Based on the symmetric expression, we only need to show that $\mathbf{p}_1$ is consistent with the eigenvector of the local transformative matrix.

To make the representation more clear, we assume $\overline{p}_1(u) > \overline{p}_2(u)$ holds for the first n_1 samples while $\overline{p}_1(u) \leq \overline{p}_2(u)$ for the others. This assumption can be verified by re-arranging the adjacent matrix and re-ordering the corresponding samples. With Eq. (5) we can separate $\mathbf{p}_1 = \begin{pmatrix} \mathbf{p}_a \\ \mathbf{0} \end{pmatrix}$ and $\overline{\mathbf{p}}_1 = \begin{pmatrix} \overline{\mathbf{p}}_a \\ \overline{\mathbf{p}}_b \end{pmatrix}$. From the RankCompete algorithm, we know that $\mathbf{p}_a$ is obtained by normalizing $\overline{\mathbf{p}}_a$. In other words,

$$\mathbf{p}_a = \rho \overline{\mathbf{p}}_a \tag{9}$$

Now we can write (7) as

$$\begin{pmatrix} \overline{\mathbf{p}}_a \\ \overline{\mathbf{p}}_b \end{pmatrix} = \mathbf{T} \begin{pmatrix} \mathbf{p}_a \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{T}_{aa} & \mathbf{T}_{ab} \\ \mathbf{T}_{ba} & \mathbf{T}_{bb} \end{pmatrix} \begin{pmatrix} \mathbf{p}_a \\ \mathbf{0} \end{pmatrix}. \tag{10}$$

From the discussion above, we can obtain

$$\begin{aligned} \overline{\mathbf{p}}_a &= \mathbf{T}_{aa} \mathbf{p}_a \\ \mathbf{p}_a &= \rho \overline{\mathbf{p}}_a, \end{aligned} \tag{11}$$

It is straightforward to see

$$\mathbf{p}_a = \rho \mathbf{T}_{aa} \mathbf{p}_a \tag{12}$$

which shows that $\mathbf{p}_a$ is the eigen vector of $\mathbf{T}_{aa}$, i.e., the ranking scores of the subgraph.

To understand better its clustering results, we next show that RankCompete can be viewed as a generalization of Normalized Cut. Normalized Cut aims to find

$$\mathbf{z}^* = \arg \min_{\mathbf{z}: z(u)=0 \text{ or } 1} \frac{\mathbf{z}^T(D-\mathbf{S})\mathbf{z}}{\mathbf{z}^T D\mathbf{z}} \tag{13}$$

which is a NP problem due to the discrete constraints. In [7], Shi and Malik relaxed problem in (13) by allowing $\mathbf{z}^*$ to take continuous value, and split $\mathbf{z}^*$ into two discrete values by choosing a splitting value.

Consider the transition matrix $\mathbf{T} = D^{-1}\mathbf{S}$, where D^{-1} is the diagonal matrix with$D(u,u) = \sum_v S(u,v)$. It is easy to see that $\mathbf{T}$ a column normalized matrix. Consider the eigen vector $\mathbf{p}$ of $\mathbf{T}$ such that $\mathbf{T}\mathbf{p} = D^{-1}\mathbf{S}\mathbf{p} = \lambda\mathbf{p}$, which can also be written as $\mathbf{S}\mathbf{p} = \lambda D\mathbf{p}$. More specifically,

$$[D-\mathbf{S}]\mathbf{p} = D\mathbf{p} - \lambda D\mathbf{p} = (I-\lambda)D\mathbf{p}$$

Based on Rayleigh quotient [7], $\mathbf{p}$ is also a solution of $\min \frac{\mathbf{p}^T(D-\mathbf{S})\mathbf{p}}{\mathbf{p}^T D\mathbf{p}}$, which means our random walk can be viewed as an iterative approximation for the Normalized Cut cost function. However, our approach is different from [7] because there is no discretization step in our method. On the contrary, we introduce two competing random walkers at the beginning and the clustering results are updated iteratively as opposed to thresholding the eigenvector in a post-processing manner.

3.2. Hierarchical RankCompete

We can also generalize RankCompete in a hierarchical fashion. The basic idea is to keep applying the two-class RankCompete in a top-down manner so that each time we split a large group into two smaller groups. The benefit of this hierarchical algorithm is that we do not need to specify the number of clusters. Instead, we use the level of hieratical clustering as parameters, and employ certain condition checks to decide whether to continue hierarchical clustering in each level. In this paper, the condition we used is whether the normalization factor ρ_k is larger than a threshold (e.g., $Threshold = 0.6$). The hierarchical structure of the clustering can help users analyze the information network with interactive action.

3.3. Discussions

Initialization: Most clustering algorithms such as K-Means and Gaussian Mixture Model are affected by the initialization. Similarly, our RankCompte might fell in non-optimal solutions with bad initializations. In practice, we first

estimate the pairwise distance by $T' \cdot T$, and then initialize starting nodes incrementally by choosing nodes which are of the smallest distance to the existing set. This intuitive strategy works well in practice. We also tried random initialization for the example in Fig 2. The results shows that for this given example, the results of RankCompete are the same no matter how we select two nodes as initialized starting points. This example shows that our RankCompete works reasonably robust in some applications.

Convergence: To analyze the convergence, we first consider the simple situation when the competition results are not changed. In this scenario, the ranking scores will guarantee to converge to the SimRank scores in the local graph. Beyond this scenario, a difficult situation when the competition results keep changing back and forth, which correspond to an oscillation state in Markov model. To cover these two scenarios, we use the following criteria to judge whether RankCompete is converged or not: (1) the ranking score $\mathbf{p}$ is no longer changed. or (2) $\mathbf{p}$ becomes similar with the values in last two iterations. To examine the convergence speed, we apply our RankCompete algorithm to a simulated network. Suppose the number of a-nodes is N and the number of s-nodes is M. We first randomly generate the adjacent matrix by $A = (rand(N, M) > 0.5)$ and then compute the transition matrix $T = AA^T$. We apply RankCompete to cluster the s-nodes into two groups. For a given size graph, we repeat the process 20 times and compare the minimum and maximum numbers of iterations with different threshold values δ. Our algorithm converges in less than 20 iterations for both $\delta = 10^{-3}$ and $\delta = 10^{-8}$. The convergence speed is not heavily affected by the threshold value. In practice, we can also analyze the computational complexity of our algorithm. Since our algorithm converges fast, we usually finish the algorithm in c iterations ($c \leq 40$ in Algorithm 1).

Limitations: Although RankCompete could provide ranking and clustering results simultaneously, it is still limited to the information embedded in network structure and overlooks each node's original features. As a results, RankCompete itself might not be able to analyze the rich social media where each subject might be associated both network structures and rich text or visual features. A compromise way is to employ multiple attribute nodes and employ the composed transition matrix by $\mathbf{T} = \sum_{l=1}^{L} \beta_l \mathbf{T}^l$, where l denotes the attribute index, and β_l is the coefficient for attribute l with the constraint of $\sum_l \beta_l = 1$. However, this formulation cannot handle those continuous features such as GPS locations or user confidences. How to design a more effective way of fusing network and node features is still an open problem.

Table 2: Comparing the clustering results on the DBLP dataset.

Models	NC[7]	NetPLSA[11]	iTopicModel[10]	RankClus[8]	RankCompete
NMI	0.6429	0.7291	0.8255	0.8390	**1.000**

4. Experiments

In this paper, we consider two kinds of information networks: bibliography network and visual information network. In our previous work [9], we show that RankCompete can be used to refine the retrieval results for web searching engines. We will present more experimental results in this section.

4.1. Bibliography Network

We use two datasets in the experiments, a four-area DBLP dataset and the Cora Research Paper Classification dataset[1]. For the DBLP data, we use the "four-area" dataset [10], which includes 20 major conferences from four related areas, i.e., database, data mining, machine learning and information retrieval, and 28702 authors and their publications in these conferences. For the Cora dataset, after preprocessing, we extracted 19396 papers with their citation lists. Each paper in Cora has a classification label from a total of 70 classes. In Cora dataset, the similarity $S(u, v)$ between two papers u and v are captured by the citation information. If u is cited by v, we let $S(u, v) = 1$; otherwise $S(u, v) = 0$.

To evaluate the clustering performance, we compare the clustering results on the DBLP dataset. Following the previous work on the bibliography network, we use the Normalized Mutual Information (NMI) measure to evaluate the clustering accuracy [8]. As Table 2 shows, our approach perfectly clusters 20 conferences from the DBLP dataset into four areas. For the more challenging Cora dataset, Table 3 shows the results on Cora dataset. We compare the performance of RankCompete and with the state-of-the-arts algorithms including Normalized Cut(NC) [7], NetPLSA [11], RankClus [8] and iTopicModel [10]. The results show that our algorithm outperforms the state of the art algorithms in clustering information networks.

Table 3: Comparing the clustering results on the Cora dataset.

Models	NC[7]	NetPLSA[11]	iTopicModel[10]	RankClus	RankCompete
NMI	0.4078	0.4291	0.4424	0.4889	**0.4945**

[1]http://www.cs.umass.edu/˜mccallum/code-data.html

The clustering results in the above tables are obtained by RankCompete in Algorithm 1. We can also apply the hierarchical RankCompete model for clustering. Figure 3 shows the results of hierarchical RankCompete, which clearly shows the correlation between the four areas. The results show that the Data Mining and Information Retrieval areas are the closest pair among the four, Database is also related, and Machine Learning is least related to them. This matches well with human expert assessment of the four concerned areas.

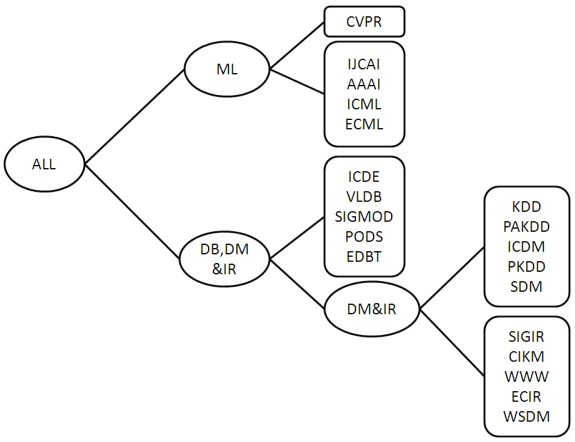

Figure 3: Clustering the DBLP author-conference dataset using hierarchical RankCompete.

To evaluate the ranking function of our RankCompete model, we use random walk scores of the conferences for conference ranking. Table 4 shows the different ranking results from PageRank and RankCompete. When using PageRank, the ranking scores are determined by the global structure, and it is often unfair to compare two conferences from different areas. For example, comparing VLDB and IJCAI is like comparing apples and oranges. Conferences from different areas often have different numbers of attendees and accept different numbers of papers, therefore it is more meaningful to compare the conferences within the same area. Our RankCompete model generates random walk scores for each cluster, and the results are more reasonable and intuitive.

We want to emphasize that our ranking results are affected by the historical factor. Since DBLP dataset contains information about authors, papers, and their venues but no citation information, our ranking is largely based on the number of papers published in a venue by the same sets of authors. Since PAKDD has a longer history than ICDM, its ranking score is higher. Due to the same reason, ICDE's ranking is higher than SIGMOD since ICDE accepts more paper from the

same group of authors. We can expect our ranking results are more consistent with human interpretation if we could use the citation information (e.g., linked with Citeseer).

Table 4: Ranking and grouping conferences.

(1) Ranking scores from PageRank

	Conferences and their ranking scores
1	VLDB(0.12) ICDE(0.12) SIGMOD(0.11) KDD(0.07) SIGIR(0.06) IJCAI(0.06) AAAI(0.06) CIKM(0.06) PODS(0.04) ICML(0.04) ICDM(0.04) EDBT(0.04) CVPR(0.04) PAKDD(0.03) WWW(0.03) SDM(0.03) PKDD(0.02) ECIR(0.02) ECML(0.01) WSDM(0.01)

(2) Ranking scores from RankCompete (clustering number = 4)

	Conferences and their ranking scores
1	IJCAI(0.35) AAAI(0.31) CVPR(0.14) ICML(0.13) ECML(0.07)
2	VLDB(0.28) ICDE(0.27) SIGMOD(0.26) EDBT(0.11) PODS(0.08)
3	KDD(0.24) PAKDD(0.24) ICDM(0.22) PKDD(0.16) SDM(0.13)
4	SIGIR(0.37) WWW(0.24) CIKM(0.23) ECIR(0.14) WSDM(0.01)

4.2. Visual Information Network

Given a collection of images, we can build a visual information network in which the images are treated as nodes, and the links are measured by the visual similarities between images. We employ the recent popular SIFT features [12] to represent local image patches, and use the number of matched patches as the similarity between a pair of images. Our RankCompete algorithm can be used for summarizing personal or web photo albums. When a user wants to view an album of his/her friends, it can be time consuming and resource stressing (e.g., from a smart phone) to view image by image in a linear fashion. To help the user review an album more efficiently, we can apply the proposed structural rank algorithm to the photo collection. As a result, photos are automatically partitioned into groups and highly ranked images in each group can be selected as representatives to produce a desirable visual summary of the album. If the user feels interested in a certain representative image, he can easily further explore the associated group of images. Such a process enables users to explore photo albums much more easily and efficiently for all conceivable scenarios.

To evaluate our approach for visual summarization, we build employ the Kodak consumer dataset [13], which is collected by handing out cameras to different users over the period of 8 months. The database consists of 17 photo collections and 1394 photos. We use hierarchical RankCompete to provide an easy way of

Figure 4: Illustration of photo album summarization. Each row corresponds a cluster in the photo album. The larger image in the left column is the representative image for each cluster, while the right column shows diverse examples of the remaining images in the cluster.

Table 5: Comparing the clustering quality of RankCompete and classical graph clustering algorithms.

Method	NC [7]	SC [6]	RankCompete
NMI score	0.39	0.42	0.44
Time (seconds)	5.61	12.31	0.89

overviewing and exploring the photo albums, of which the level parameter is 3, and the number of clusters for normalized cut and spectral clustering is select as $2^3 = 8$. The size of these collections varies from 22 to 194, which reflects how the photographers separate the images naturally according to the events. In this experiment, the similarity between two images is computed as the shared similar local patches.

To evaluate our clustering algorithm, we benchmark against the ground truth set by the photographer who took the photos. Since the bibliography clustering algorithms [8] [10] are not designed for this problem, we compare RankCompete with normalized cut [7] and Ng's spectral clustering algorithms [6]. As shown in Table 5, our algorithm outperforms normalized cut (NC) and spectral clustering (SC). As we discussed in Section 3.1, our model can be viewed as a generalization of normalized cut, by replacing the discretization step in NC with a competition process. The results shows that the new approach works for photo album data.

Since our RankCompete provide not only clustering but also ranking results, it could be used to design a new interface to explore the personal photos. Figure 4 shows some of the album summarization results using RankCompete. The left-hand column contains the representative images for different clusters, while the right-hand column contains other images in the corresponding clusters. Qualitatively, the left-hand column of representative images generally provides a more

concise view of the diverse content in a personal photo album. We further expect that the quality of the summarization to improve with richer visual features and contextual cues in addition to SIFT features.

5. Acknowledgement

We would like thank anonymous reviewers for their comments. We are thankful to Yizhou Sun for her kind help with our experimental comparison. This research was sponsored in part by the U.S. National Science Foundation under grants IIS-1049332 EAGER, CCF-0905014, CNS-0931975, CNS-1027965, and IIS-0713581, by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053 (NS-CTA), and Air Force Office of Scientific Research MURI award FA9550-08-1-0265.

6. Conclusions

In this paper, we introduce the concept of "competition" into the random walk framework and develop a new model called RankCompete. Our new model can do clustering and ranking simultaneously in such a way that the two tasks help each other for better performance. We validate our algorithms on the paper-conference network and visual information network. Our future work will be to apply RankCompete algorithm for large scale information network analysis.

Reference

[1] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web, Stanford Digital Libraries.

[2] J. Kleinberg, Authoritative sources in a hyperlinked environment, JACM 46 (5) (1999) 604–632.

[3] K. Bharat, G. Mihaila, Hilltop: A search engine based on expert documents, in: WWW, 2000.

[4] T. Haveliwala, Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search, IEEE transactions on knowledge and data engineering (2003) 784–796.

[5] D. Zhou, O. Bousquet, T. Lal, J. Weston, B. Scholkopf, Learning with local and global consistency, NIPS 16 (2004) 321–328.

[6] A. Y. Ng, M. I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: NIPS, 2002.

[7] J. Shi, J. Malik, Normalized cuts and image segmentation, PAMI 22 (8) (2000) 888–905.

[8] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, T. Wu, Rankclus: Integrating clustering with ranking for heterogenous information network analysis, in: EDBT, Vol. 9, 2009.

[9] L. Cao, A. D. Pozo, X. Jin, J. Luo, J. Han, T. S. Huang, Rankcompete: Simultaneous ranking and clustering of web photos, in: WWW, 2010.

[10] Y. Sun, J. Han, J. Gao, Y. Yu, iTopicModel: Information Network-Integrated Topic Modeling, ICDM (2009) 493–502.

[11] Q. Mei, D. Cai, D. Zhang, C. Zhai, Topic modeling with network regularization, in: WWW, 2008.

[12] D. Lowe, Object recognition from local scale-invariant features, in: ICCV, 1999.

[13] L. Cao, J. Luo, T. Huang, Annotating photo collections by label propagation according to multiple similarity cues, in: ACM Conf. Multimedia, 2008.