BiRank: Towards Ranking on Bipartite Graphs

Xiangnan He, Ming Gao, *Member, IEEE*, Min-Yen Kan, *Member, IEEE*, and Dingxian Wang

Abstract—The bipartite graph is a ubiquitous data structure that can model the relationship between two entity types: for instance, users and items, queries and webpages. In this paper, we study the problem of ranking vertices of a bipartite graph, based on the graph's link structure as well as prior information about vertices (which we term a *query vector*). We present a new solution, BiRank, which iteratively assigns scores to vertices and finally converges to a unique stationary ranking. In contrast to the traditional random walk-based methods, BiRank iterates towards optimizing a regularization function, which smooths the graph under the guidance of the query vector. Importantly, we establish how BiRank relates to the Bayesian methodology, enabling the future extension in a probabilistic way. To show the rationale and extendability of the ranking methodology, we further extend it to rank for the more generic n-partite graphs. BiRank's generic modeling of both the graph structure and vertex features enables it to model various ranking hypotheses flexibly. To illustrate its functionality, we apply the BiRank and TriRank (ranking for tripartite graphs) algorithms to two real-world applications: a general ranking scenario that predicts the future popularity of items, and a personalized ranking scenario that recommends items of interest to users. Extensive experiments on both synthetic and real-world datasets demonstrate BiRank's soundness (fast convergence), efficiency (linear in the number of graph edges), and effectiveness (achieving state-of-the-art in the two real-world tasks).

Index Terms—Bipartite graph ranking, graph regularization, n-partite graphs, popularity prediction, personalized recommendation

1 Introduction

Graphs provide a universal language to represent relationships between entities. In real-world applications, not only should the relationships between entities of the same type be considered, but the relationships between different types of entities should also be modeled. Such relationships naturally form a bipartite graph, containing rich information to be mined from. For example, in YouTube, the videos and users form a bipartite relationship where edges indicate a viewing action; in Web search, the relationships between queries and search engine result pages are user actions ("clicks"), which provide important relevance judgments from the user's perspective.

A fundamental task in the mining of bipartite graphs is to rank vertices against a specific criterion. Depending on the setting, assigning each vertex a ranking score can be used for many tasks, including the estimation of vertex importance (popularity prediction) and the inference of similar vertices to a target vertex (similarity search), and edge suggestion for connecting a target vertex (link prediction and recommendation). Existing work on graph ranking have largely focused on unipartite graphs, including PageRank [2], HITS [3],[1] and many of their variants [4], [5], [6], [7]. Although several works [8], [9], [10] have considered ranking on bipartite graphs, they have either focused on a specific application or adapted existing algorithms to handle the bipartite case. In our opinion, the work up to the current time, lacks a thorough theoretical analysis.

In this paper, we focus on the problem of ranking vertices of bipartite graphs. We formulate the ranking problem in a generic manner—accounting for both the graph's structural information and the proper incorporation of any prior information for vertices, where such vertex priors can be used to encode any features of vertices. The main contributions of this paper are summarized as follows:

- We develop a new algorithm—BiRank—for addressing the ranking problem on bipartite graphs, and show its convergence to a unique stationary point;
- We analyze BiRank through the formalism of graph regularization, and present a complementary Bayesian view. These two views enable future extensions to be grounded and compelling from a theoretically principled way (either algebraically or probabilistically).
- We deploy BiRank to the general ranking scenario of item popularity prediction, illustrating how to parameterize it to encode several ranking hypotheses;
- We extend the methodology to rank on the more generic n-partite graphs, and employ it for a personalized ranking scenario by mining tripartite graphs.
- We conduct extensive experiments to justify our methods for the two real-world ranking scenarios of popularity prediction and personalized recommendation.

The paper is organized as follows. After reviewing related works in Section 2, we formalize the problem in Section 3. Then we describe the BiRank algorithm in Section 4, and interpret it from two views in Section 5. In Section 6, we discuss how to apply BiRank to popularity prediction and

1. Note that although HITS does handle bipartite graphs, the algorithm was designed for ranking on unipartite graphs by treating vertices with two roles—hub and authority.

- *X. He and M.-Y. Kan are with the Web IR/NLP Group, School of Computing, National University of Singapore, Singapore 11907. E-mail: xiangnanhe@gmail.com, kanmy@comp.nus.edu.sg.*
- *M. Gao is with the Software Engineering Institute, East China Normal University, Shanghai 200062, China. E-mail: mgao@sei.ecnu.edu.cn.*
- *D. Wang is with the Ranking team, Search Science Department, eBay Inc., Shanghai 200040, China. E-mail: diwang@ebay.com.*

Manuscript received 27 Jan. 2016; revised 19 Aug. 2016; accepted 13 Sept. 2016. Date of publication 20 Sept. 2016; date of current version 5 Dec. 2016. Recommended for acceptance by Y. Chang.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2016.2611584

personalized recommendation. We conduct experiments in Section 7, before concluding the paper in Section 8.

2 Related Work

BiRank, which ranks vertices of a bipartite graph, can be categorized as a link-based object ranking method under the paradigm of link mining [11]. In this section, we focus on related work that contribute in the ranking method, and omit discussion of other relevant issues such as efficiency and evolving graphs. We then review work that can benefit from such bipartite graph ranking, forming the potential downstream applications of BiRank.

2.1 Graph Ranking Methods

In the context of web graph ranking, PageRank [2] and HITS [3] are the most prominent methods. PageRank estimates the importance score of vertices as the stationary distribution of a random walk process—starting from a vertex, the surfer randomly jumps to a neighbor vertex according to the edge weight. HITS assumes each vertex has two roles: *hub* and *authority*, transforming the graph to a bipartite graph. A vertex has a high authority score if it is linked by many vertices with hub score, and a vertex has a high hub score if it links to many authoritative vertices.

Many variants start from the basic themes of PageRank and HITS. Ng et al. [12] studied the stability of the two algorithms, finding HITS more sensitive to small perturbations in the graph structure under certain situations. They proposed two variants—Randomized HITS and Subspace HITS—that yield more stable rankings. Similarly, Lempel et al. [5] found that applying HITS on graphs with TKCs (*tightly knit communities*, i.e., small but highly interconnected set of vertices) fails to identify meaningful authority vertices. They devised SALSA as a stochastic variant of HITS, for alleviating the TKC effect. Haveliwala [4] proposed topic-sensitive PageRank (also known as *personalized PageRank*) by replacing the uniform teleportation vector with a non-uniform vector that encodes each vertex's topic score (cf. query vector in our BiRank context). Later on, Ding et al. [13] unified HITS and PageRank under a normalized ranking framework. Inspired by the discrete-time Markov process explanation of PageRank, Liu et al. [6] also proposed BrowseRank based on continuous time Markov process, exploiting user behavior data for page importance ranking. To incorporate side information on vertices and edges into ranking, Gao et al. [7] extended PageRank in a semi-supervised way by learning the transition matrix based on the features on vertices and edges.

Along a separate line of work—ranking on graphs based on regularization theory [14], [15], [16]—has gained popularity within the machine learning community. These works mainly consider the problem of labeling vertices of a graph from partially known labels, also termed *semi-supervised learning* or *manifold learning* on graphs. Smola et al. [15] summarized early works on graph kernels (e.g., Diffusion kernels), and formulated a family of regularization operators on graphs to encompass such kernels. Inspired by it, Zhou et al. [14] developed a regularization framework consisting of two constraints: smoothness and fitting, proposing an iterative algorithm [17] for optimizing the regularization function. Later on, they et al. supplemented the regularization framework by developing a discrete analytic theory of graphs [18] and extending it to cover directed graphs [19]. Agarwal [16] further extended the regularization framework by replacing the fitting term (i.e., sum of squared errors) to the *hinge* ranking loss, proposing an algorithm with similarities to solving *support vector machine* to optimize the regularization function.

The above discussed works have all focused on ranking for homogeneous graphs, where vertices are of the same type. Our proposed BiRank targets the task of ranking for bipartite graphs, where vertices are of two different types. Separately handling the two vertex types is very important for ranking in bipartite graphs for many applications, a claim we validate through our experiments later. Inspired by the graph regularization framework [18], we develop the BiRank algorithm, which can be seen an extension of the manifold ranking algorithm [17] on bipartite graphs.

2.2 Ranking on Bipartite Graphs

There are other algorithms developed for bipartite graph ranking that target specific applications. As a natural way to represent relationship between two types of entities, bipartite graphs have been widely used across domains. As a consequence, ranking on bipartite graph data have been explored to address many applications. For example, in Web search, Deng et al. [8] modeled queries and URLs for query suggestion, Cao et al. [20] considered the co-occurrence between entities and queries for entity ranking, Li et al. [10] modeled users and their search sessions for detecting click spam, and Rui et al. [21] mined visual features and the surrounding texts for Web image annotation. In practical recommender systems, bipartite graphs methods have been used for Twitter user recommendation [22] and YouTube video recommendation [23]. In the domain of natural language processing, Parveen et al. [24] generated multi-document summarization based on the relationship of sentences and lexical entities.

In terms of the ranking technique, these works share the same cornerstone—they all rank by iteratively propagating scores on the graph; either through a PageRank-like random walk or a HITS-like iterative process—which is adjusted for use on bipartite graphs. The prominent advantage of such propagation-based methods is that the global structure of the graph can be implicitly considered, which is an effective way to deal with the data sparsity and make use of the graph structure. Similar to their ranking algorithms, our proposed BiRank is also a propagation-based method; however, the main difference lies in the normalization strategy used in the iterative process. The symmetric normalization used in BiRank normalizes an edge weight by both of its vertex ends, which accords a smoothing on the graph that can be explained by the regularization theory [18]. As a result, extensions to the algorithm, such as incorporating more features about vertices and edges, can be achieved in a theoretically principled way. More importantly, we believe such a bridge with the graph regularization theory and Bayesian framework allows BiRank a broader algorithmic extensions that are difficult to achieve by PageRank, HITS and their variants. For example, we can adjust the

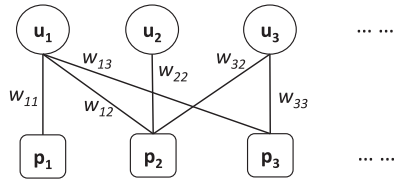


Fig. 1. Bipartite user-item structure.

score propagation process to use a different ranking-based objective (Section 5.1.2), and learn the combination parameters in an automatic way (Section 5.2). We will study these algorithmic extensions of BiRank in the future.

3 Problem Formulation

We first present the bipartite graph model and then give the notation convention used. We then formalize the ranking problem that we address in this paper.

Notations. Let $G = (U \cup P, E)$ be a bipartite graph, where U and P represent the vertex sets of the two entity types respectively, and E represents the edge set (*n.b.*, bipartite graphs have edges only between vertices of the two different types). Fig. 1 shows an example of the bipartite structure.

We use u_i to denote the ith vertex in U, and p_j to denote the jth vertex in P, where $1 \leq i \leq |U|$ and $1 \leq j \leq |P|$; set cardinality $|U|$ denotes number of elements in U. Edges carry non-negative weights w_{ij}, modeling the relationship strength between the connected vertices u_i and p_j (if u_i and p_j are not connected, their edge weight w_{ij} is zero). As such, we can represent all edge weights of the graph as a $|U| \times |P|$ matrix $W = [w_{ij}]$. For each vertex u_i, we denote its weighted degree (i.e., sum of connected edges' weights) as d_i, and use a diagonal matrix D_u to denote the weighted degrees of all vertices in U such that $(D_u)_{ii} = d_i$; and similarly, for d_j and D_p. Note that in this paper, we deal with undirected bipartite graphs, i.e., we do not model any directionality in the edges.

Problem Definition. In a nutshell, the general graph ranking problem is to assign each vertex a score *s.t.*a given expectation is satisfied. For example, PageRank [2] infers an importance score for each vertex to capture the intuition that an important vertex should be linked by many other important vertices. As in many applications, a ranking simply based on the graph structure is insufficient; often, there also exists some prior information (or features) on the vertices. For example, in webpage ranking, we already know some webpages are important (e.g., official sites), and wish to incorporate this knowledge into the ranking process; in the application of recommendation, we need to consider a user's historical actions as the prior knowledge of the user's preference. We term such prior knowledge as a *query vector*, which encodes the prior belief of the score of vertices with respect to the ranking criterion. In this paper, we study the bipartite graph ranking problem where a query vector is given, formally defined as:

Input: A bipartite graph $G = (U \cup P, E)$ with its weight matrix W. A query vector $\mathbf{u}^0, \mathbf{p}^0$ encodes the prior belief concerning the vertices in U and P, respectively, with respect to the ranking criterion.

Output: A function $f : P \cup U \to \mathbb{R}$, which maps each vertex in G to a real number.

The function value $f(u_i)$ and $f(p_j)$ form the ranking score of vertex u_i and p_j, respectively. To keep the notation simple, we also use u_i and p_j to denote the ranking score, and represent the final ranking score of all vertices as two ranking vectors $\mathbf{u} = [u_i]$ and $\mathbf{p} = [p_j]$.

4 Iterative BiRank

With the preliminaries settled, we now detail the iterative paradigm of the BiRank algorithm. We first describe how we design the ranking algorithm, analyzing its time complexity. Then we study its convergence properties in theory. Finally we discuss the connection of BiRank with other similarly-styled iterative bipartite graph ranking algorithms.

4.1 BiRank's Design

To rank vertices based on the graph structure, seminal algorithms like PageRank and HITS have been proposed. Motivated from their design, our intuition for bipartite graph ranking is that the scores of vertices should follow a smoothness convention, namely that: *a vertex (from one side) should be ranked high if it is connected to higher-ranked vertices (from the other side)*. This rule defines a mutually-reinforcing relationship, which is naturally implemented as an iterative process that refines each vertex's score as the sum of the contribution from its connected vertices

$$p_j = \sum_{i=1}^{|U|} w_{ij} u_i; \quad u_i = \sum_{j=1}^{|P|} w_{ij} p_j.$$

As it is an additive update rule, normalization is necessary to ensure the convergence and stability. Two strategies have been widely adopted in previous work: 1) a PageRank-style that normalizes W (and W^T) to a stochastic matrix, leading to a probabilistic random walk explanation; and 2) a HITS-style method that normalizes the ranking scores of vertices after each iteration. In our BiRank method, we adopt the symmetric normalization scheme, which is inspired from Zhou et al.'s work [14] addressing semi-supervised learning on graphs. The idea is to smooth an edge weight by the degree of its two connected vertices simultaneously

$$p_j = \sum_{i=1}^{|U|} \frac{w_{ij}}{\sqrt{d_i}\sqrt{d_j}} u_i; \quad u_i = \sum_{j=1}^{|P|} \frac{w_{ij}}{\sqrt{d_i}\sqrt{d_j}} p_j, \tag{1}$$

where d_i and d_j are the weighted degrees of vertices u_i and p_j, respectively. The use of symmetric normalization is a key characteristic of BiRank, allowing edges connected to a high-degree vertex to be suppressed through normalization, lessening the contribution of high-degree vertices. This has the beneficial effect of toning down the dependence of top rankings on high-degree vertices, a known defect of the random walk-based diffusion methods [23]. This gives rise to better quality results.

To account for the query vector $\mathbf{p}^0$ and $\mathbf{u}^0$ that encode the prior belief on the importance of the vertices, one can either opt for 1) incorporating the graph ranking results for combination in post-processing (*a.k.a* late

fusion), or 2) factoring the query vector directly into the ranking process. The first way of post-processing yields a ranking that is a compromise between two rankings; for scenarios that the query vector defines a full ranking of vertices, this ensemble approach might be suitable. However, when the query vector only provides *partial* information—i.e., only a small proportion of vertices have a prior score while most other vertices have no prior information—this method fails to identify an optimal ranking. For example, in the application of personalized recommendation (see Section 6.2), the aim is to rank *unconsumed items* for a user; the query vector encodes the user's known preference, which is a sparse vector with the consumed items as non-zeros. In this case, simply combining the ranking from graph structure and query vector via post processing does not work, since the ranking of *unconsumed items* will solely depend on the graph structure. As such, in BiRank we opt for the second way that factors the query vector directly into the ranking process, which has the advantage of using the query vector to guide the ranking process

$$\begin{aligned} p_j &= \alpha \sum_{i=1}^{|U|} \frac{w_{ij}}{\sqrt{d_i}\sqrt{d_j}} u_i + (1-\alpha) p_j^0; \\ u_i &= \beta \sum_{j=1}^{|P|} \frac{w_{ij}}{\sqrt{d_i}\sqrt{d_j}} p_j + (1-\beta) u_i^0, \end{aligned} \tag{2}$$

where α and β are hyper-parameters to weight the importance of the graph structure and the prior query vector, to be set between $[0, 1]$. To keep notation simple, we can also express the iteration in its equivalent matrix form

$$\begin{aligned} \mathbf{p} &= \alpha S^T \mathbf{u} + (1-\alpha)\mathbf{p}^0; \\ \mathbf{u} &= \beta S \mathbf{p} + (1-\beta)\mathbf{u}^0, \end{aligned} \tag{3}$$

where $S = D_u^{-\frac{1}{2}} W D_p^{-\frac{1}{2}}$, the symmetric normalization of weight matrix W. We call this set of update rules the *BiRank iteration*, which forms the core of the iterative BiRank algorithm. In a nutshell, BiRank first randomly initializes the ranking vector, and then iteratively executes the BiRank iteration until convergence (summarized in Algorithm 1).

Algorithm 1. The Iterative BiRank Algorithm

Input: Weight matrix W, query vector $\mathbf{p}^0, \mathbf{u}^0$, and hyper-parameters α, β;
Output: Ranking vectors $\mathbf{p}, \mathbf{u}$;
1: Symmetrically normalize W:$S = D_u^{-\frac{1}{2}} W D_p^{-\frac{1}{2}}$;
2: Randomly initialize $\mathbf{p}$ and $\mathbf{u}$;
3: **while** Stopping criteria is not met **do**
4: $\quad \mathbf{p} \leftarrow \alpha S^T \mathbf{u} + (1-\alpha)\mathbf{p}^0$;
5: $\quad \mathbf{u} \leftarrow \beta S \mathbf{p} + (1-\beta)\mathbf{u}^0$;
6: **end**
7: **return** $\mathbf{p}$ and $\mathbf{u}$

For convergence, one can either monitor the change of ranking vectors $\mathbf{p}, \mathbf{u}$ across iterations, or rely on a held-out validation data to prevent overfitting. Moreover, the numerical convergence of BiRank is theoretically guaranteed, discussed later in Section 4.2.

4.1.1 Time Complexity Analysis

It is easy to show that a direct implementation of BiRank iteration in Eq. (3) has a time complexity of $O(|P| \cdot |U|)$, mainly due to the multiplication of $S^T\mathbf{u}$ and $S\mathbf{p}$. However, note that in real-world applications, the matrix S is typically very sparse; for example in recommender systems, the user-item matrix to model is always over 99 percent sparse (e.g., Netflix challenge dataset). In this case, a representation of sparse matrix only needs to account for non-zero entries (which correspond to the edges of the bipartite graph), instead of all $|P| \cdot |U|$ entries. As such, the real-time cost needed for BiRank is $O(c|E|)$, where c denotes the number of iterations executed to converge, and $|E|$ denotes number of edges in the graph. Thus, BiRank is linear with respect to number of edges, ensuring good scalability to large-scale graphs. Moreover, our empirical experience show that BiRank has a very fast convergence rate—10 iterations are usually enough for convergence. One reason is that it can be seen as optimizing a convex function effectively using alternating optimization, discussed later in Section 5.

4.2 Convergence Analysis of BiRank

We show that BiRank can converge to a stationary and unique solution regardless of the initialization, followed by a theoretical analysis of the convergence speed.

4.2.1 Proof of Convergence

It is clear that the behavior of BiRank depends on the hyper-parameters α and β, which are in the range $[0, 1]$. To make a through analysis, we need to carefully consider the boundary conditions. Considering the two boundaries 0 and 1, we divide the proof into the following three cases:

Proof. 1. $\alpha = 0$ or $\beta = 0$. When $\alpha = 0$, the vector $\mathbf{p} = \mathbf{p}^0$ is unchanged across iterations. Thus $\mathbf{u}$, which depends on $\mathbf{p}$ and $\mathbf{u}^0$, will also be unchanged after the first iteration. Similarly for the case of $\beta = 0$.

2. $\alpha = 1$ and $\beta = 1$. In this case, the query vectors do not have any impact on the ranking, and the ranking is solely determined by the graph structure. The iterative update rule then reduces to Eq. (1), whose matrix form is $\mathbf{p} = S^T\mathbf{u}, \mathbf{u} = S\mathbf{p}$. By further reducing this, we obtain

$$\begin{aligned} \mathbf{p}^{(k)} &= (S^T S)\mathbf{p}^{(k-1)} = \cdots = (S^T S)^k \mathbf{p}^{(0)}, \\ \mathbf{u}^{(k)} &= (S S^T)\mathbf{u}^{(k-1)} = \cdots = (S S^T)^k \mathbf{u}^{(0)}, \end{aligned} \tag{4}$$

where k denotes the number of iterations, and $\mathbf{p}^{(0)}$, $\mathbf{u}^{(0)}$ denote the initial ranking scores for vertices. Note that matrix $S^T S$ and SS^T are both symmetric matrices. According to a lemma in standard linear algebra [25]. □

Lemma 1. *If M is a symmetric matrix, and v is a vector not orthogonal to the principle eigenvector of M, then the limit of $M^k v$ (after scaling to a unit vector) converges to the principle eigenvector of M with k increasing without bound.*

By the lemma, we can see that with reasonable initialization, the iterative process will converge to a stationary solution $\mathbf{p}^*$ and $\mathbf{u}^*$, which are the principle eigenvector of matrix $S^T S$ and SS^T, respectively.

3. Normal cases. We now consider the normal ranking scenarios that α and β are in the range of (0,1), or one of α, β is 1, meaning that both the graph structure and query vectors can affect the ranking process. Without loss of generality, we prove the convergence of $\mathbf{p}$.

First, we eliminate $\mathbf{u}$ in $\mathbf{p}$'s update rule

$$\mathbf{p} = \alpha\beta(S^T S)\mathbf{p} + \alpha(1-\beta)S^T\mathbf{u}^0 + (1-\alpha)\mathbf{p}^0. \quad (5)$$

Let matrix M be $\alpha\beta(S^T S)$ and vector $\mathbf{z}_0$ be $\alpha(1-\beta)S^T\mathbf{u}^0 + (1-\alpha)\mathbf{p}^0$, which are both invariant across iterations. Then, we have

$$\mathbf{p}^{(k)} = M\mathbf{p}^{(k-1)} + \mathbf{z}_0 = \cdots = M^k\mathbf{p}^{(0)} + \sum_{t=0}^{k-1} M^t\mathbf{z}_0, \quad (6)$$

where k denotes the number of iterations, and $\mathbf{p}^{(0)}$ denotes the initial ranking vector of $\mathbf{p}$. Assuming M's eigenvalues are in the range of $(-1, 1)$, we can obtain

$$\lim_{k\to\infty} M^k\mathbf{p}^{(0)} = 0, \text{and } \lim_{k\to\infty}\sum_{t=0}^{k-1} M^t = (I-M)^{-1},$$

where I denotes the identity matrix. In this case, we can derive the stationary solution of $\mathbf{p}$ as

$$\mathbf{p}^* = (I-M)^{-1}\mathbf{z}_0. \quad (7)$$

However, the above stationary solution is derived based on the assumption that M's eigenvalues are in the range $(-1,1)$. Next, we prove the correctness of this assumption.

Theorem 1. *M's eigenvalues are in the range $[-\alpha\beta, \alpha\beta]$.*

Proof. Recall that M is defined as

$$M = \alpha\beta(S^T S) = \alpha\beta\left(D_u^{-\frac{1}{2}}WD_p^{-\frac{1}{2}}\right)^T\left(D_u^{-\frac{1}{2}}WD_p^{-\frac{1}{2}}\right) = \alpha\beta\left(D_p^{-\frac{1}{2}}W^T D_u^{-1}WD_p^{-\frac{1}{2}}\right).$$

To see M's eigenvalues are bounded by $\alpha\beta$, we first construct a variant M_v that has the same eigenvalues[2] as M

$$M_v = D_p^{\frac{1}{2}}MD_p^{-\frac{1}{2}} = \alpha\beta(W^T D_u^{-1}WD_p^{-1}).$$

Note that matrix $W^T D_u^{-1}WD_p^{-1}$ is a stochastic matrix in which the entries of each column sum to 1. By standard linear algebra [25], for a stochastic matrix, its largest absolute value of the eigenvalues is always 1. Thus, the eigenvalues of M_v are in the range $[-\alpha\beta, \alpha\beta]$, and same must hold for M as they have exactly the same eigenvalues. The proof of $M's$ eigenvalues is finished. □

As M's eigenvalues are theoretically guaranteed in the range $[-\alpha\beta, \alpha\beta]$ and in the normal cases α, β are in the range $(0, 1)$, the assumption that M's eigenvalues are in the range $(-1, 1)$ holds. Therefore, we conclude that Eq. (7) indeed forms the stationary solution of $\mathbf{p}$. To round out the proof, we give the stationary solution of BiRank as follows:

$$\begin{aligned}\mathbf{p}^* &= (I - \alpha\beta S^T S)^{-1}[\alpha(1-\beta)S^T\mathbf{u}^0 + (1-\alpha)\mathbf{p}^0],\\ \mathbf{u}^* &= (I - \alpha\beta SS^T)^{-1}[\beta(1-\alpha)S\mathbf{p}^0 + (1-\beta)\mathbf{u}^0].\end{aligned} \quad (8)$$

The convergence proof of BiRank is finished.

This convergence proof gives an elegant closed-form solution—for any non-trivial initializations, the iterative algorithm of BiRank will converge to Eq. (8). As such, an alternative method to our iterative BiRank is to direct calculate the closed-form stationary solution. Even so, we suggest the practitioners following the iterative procedure for two reasons. First, in real-world practice, when there is a large number of vertices to rank, the matrix inversion operation is very expensive, making the calculation of the closed-form solution inefficient. More specifically, matrix inversion is usually assumed to take $O(N^3)$ time [26]; thus the time complexity of directly calculating the stationary solution is $O(|P|^3 + |U|^3)$, which even can be much higher than the upper bound of the iterative solution $O(c|P||U|)$. Second, the iterative process emphasizes the underlying motivation that reinforcing a vertex's importance from its neighbors and the query vector. As such, one does not have to run the iterations until convergence; instead, one can compute the scores by starting from any initialization and performing a fixed number of iterations.

4.2.2 Speed of Convergence

Since the behavior of BiRank depends on the graph structure, query vector and hyper-parameters α, β, we analyze how do these factors impact BiRank's convergence speed.

In each BiRank iteration, the score of a vertex comes from both its neighbors and the query vector. Since the query vector is static that remains unchanged across iterations, it cannot contribute to any form of divergence in the rankings; thus the main uncertainty for convergence stems from the part of the score diffusion from neighbors. As such, the number of iterations required to converge will increase as α and β increase (the empirical evidence in Fig. 6 also verifies this property). Clearly, the slowest convergence is when α and β are set to 1, where the effect of query vector is eliminated. When both α and β are set to 1, the update of $\mathbf{p}$ at the iteration k can be written as: $\mathbf{p}^{(k)} = (S^T S)\mathbf{p}^{(k-1)}$, which essentially can be seen as the power method for the symmetric eigenvalue problem (Chapter 8.2 of [25]). It is known that the convergence of power method is determined by the second dominant eigenvalue of the transition matrix. In spite of the slight difference that the power iteration requires an additional L_2 normalization on the ranking vector (while our BiRank does not), we point out that BiRank shares the same property of convergence speed.

Theorem 2. *The convergence rate of BiRank depends on the second largest eigenvalue of the matrix $S^T S$ in magnitude.*

Proof. As $S^T S$ is a symmetric matrix, it is guaranteed to have n eigenvalues which are real numbers ($n = |P|$). Let its eigenvalues be $\lambda_1, \lambda_2, \ldots, \lambda_n$, where $|\lambda_1| \geq |\lambda_2| \geq, \ldots, \geq |\lambda_n|$, and vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ be the corresponding Eigenvectors. Then, the starting vector $\mathbf{p}^{(0)}$ can be expressed as: $\mathbf{p}^{(0)} = \sum_{i=1}^n c_i\mathbf{x}_i$, where $\{c_i\}$ are constant coefficients. Then

2. The equality of the eigenvalues is easily shown by using determinants, denoted as $|\cdot|$. Let the eigenvalues of M_v be λ_v, then we have $|M_v - \lambda_v I| = |D_p^{\frac{1}{2}}(M - \lambda_v I)D_p^{-\frac{1}{2}}| = |D_p^{\frac{1}{2}}| \cdot |M - \lambda_v I| \cdot |D_p^{-\frac{1}{2}}| = |M - \lambda_v I| = 0$, meaning that λ_v are also M's eigenvalues.

TABLE 1
Transition Matrices (i.e., S and S^T in Eq. (3)) of Different Bipartite Graph Ranking Algorithms

Method	Definition of Transition Matrices
HITS (Kleinberg [3])	$S = W; S^T = W^T$
Co-HITS (Deng et al. [8])	$S = WD_p^{-1}; S^T = W^T D_u^{-1}$
BGER (Cao et al. [20])	$S = D_u^{-1}W; S^T = D_p^{-1}W^T$
BGRM (Rui et al. [21])	$S = D_u^{-1}WD_p^{-1}; S^T = D_p^{-1}W^T D_u^{-1}$
BiRank (our proposal)	$S = D_u^{-\frac{1}{2}}WD_p^{-\frac{1}{2}}; S^T = D_p^{-\frac{1}{2}}W^T D_u^{-\frac{1}{2}}$

Note that S^T here denotes a matrix, rather than just the transpose of S.

the update of $\mathbf{p}^{(k)}$ can be written as

$$\begin{aligned}\mathbf{p}^{(k)} &= c_1(S^TS)^k\mathbf{x}_1 + c_2(S^TS)^k\mathbf{x}_2 + \cdots + c_n(S^TS)^k\mathbf{x}_n \\ &= c_1\lambda_1^k(x_1 + \sum_{i=2}^{n} c_i(\lambda_i/\lambda_1)^k\mathbf{x}_i).\end{aligned} \tag{9}$$

Here we use the fact that $(S^TS)\mathbf{x}_i = \lambda_i\mathbf{x}_i$. Hence, we see that the non-essential quantities decay at a rate of approximately $|\lambda_2/\lambda_1|$. As we have shown in Theorem 1, S^TS has the same eigenvalues with a variant stochastic matrix, thus we have $|\lambda_1| = 1$. The proof is finished. □

To summarize, the convergence rate of BiRank depends on the normalized adjacency matrix S and parameters α, β. Analytically, larger α and β will lead to slower convergence; theoretically, smaller magnitude of the second dominant eigenvalue of S^TS will result in faster convergence. In many applications, for high dimensional but sparse relational data (e.g., user behaviors, documents), S is usually of low rank. As a result, $|\lambda_2|$ is a small number, leading to a fast convergence of our BiRank algorithm.

4.3 Connection with Other Algorithms

There are some bipartite graph ranking algorithms [8], [20], [21] that share a similar spirit with BiRank, though originally developed for specific applications with varying ranking targets. Specifically, in terms of the iterative ranking process, they have the same update rule form as Eq. (3); the main difference is in how to generate the transition matrices (S and S^T for updating $\mathbf{u}$ and $\mathbf{p}$, respectively). It is instructive to clarify the difference with these algorithms.

Table 1 summarizes the ways of constructing transition matrices using our symbol notation. From a high-level view, these algorithms differ in how they utilize the vertex degree to normalize each edge weight (except that HITS does not account for the query vector). HITS, the earliest proposed algorithm, uses the original weight matrix W as-is; although the convergence can be guaranteed in theory, HITS is sensitive to outliers in graph [12] and suffers from the tightly knit communities phenomenon [5]. Co-HITS [8] normalizes each column of W (and W^T) stochastically, having an explanation of simulating random walks on the graph. However, random walk methods can be biased towards the high-degree vertices [23]. While BGER [20] avoids this defect by normalizing each row of W (and W^T) stochastically, yielding an effect of suppressing the scores of high-degree vertices. However, the one-side normalization of BGER does not account the degrees of $\mathbf{p}$ vertices when updating $\mathbf{u}$, allowing high-degree $\mathbf{p}$ vertices to exert a stronger impact in the diffusion process; and vice versa. Similar with our proposed BiRank, BGRM also applies a symmetric normalization on W, while the level of normalization differs (the sum of normalization exponents is -2 and -1 for BGRM and BiRank, respectively). Although it is difficult to tell which way between them is more advantageous, we point out that BiRank employs the matrix S^TS in a similar fashion to a stochastic matrix (the same eigenvalues, see Theorem 1) and corresponds to a regularization framework, both of which are nice properties that BGRM lacks.

5 Foundations of BiRank

In contrast to the traditional graph ranking algorithms (e.g., PageRank and HITS), BiRank iterations are implicitly optimizing an objective function. This is analogous to the manifold ranking algorithm on graphs [14]. In what follows, we investigate the regularization framework for BiRank and present a Bayesian explanation of the ranking algorithm. These two views shed important insight into the basis of BiRank, allowing future extensions in a theoretically principled way. To show its extendability, we finally generalize the methodology to rank for the more general n-partite graphs.

5.1 Regularization Framework

Inspired from the discrete graph theory [18], we construct the regularization function as follows:

$$\begin{aligned}R(\mathbf{u},\mathbf{p}) = &\sum_{j=1}^{|P|}\sum_{i=1}^{|U|} w_{ij}\left(\frac{p_j}{\sqrt{d_j}} - \frac{u_i}{\sqrt{d_i}}\right)^2 \\ &+ \gamma\sum_{j=1}^{|P|}(p_j - p_j^0)^2 + \eta\sum_{i=1}^{|U|}(u_i - u_i^0)^2,\end{aligned} \tag{10}$$

where γ and η are the regularization parameters to combine different components (they are constants corresponding to α and β in BiRank). Next, we first show that optimizing Eq. (10) leads to the iterative BiRank algorithm, and then interpret the meaning of the regularization function.

5.1.1 Relationship with BiRank

Eq. (10) defines an objective function with the ranking scores as model parameters. To optimize the objective function, a common strategy is performing the coordinate descent [27]. Let us first calculate its first-order derivatives with respect to each model parameter

$$\begin{aligned}\frac{\partial R}{\partial p_j} &= (2+2\gamma)p_j - 2\gamma p_j^0 - 2\sum_{i=1}^{|U|}\frac{w_{ij}u_i}{\sqrt{d_i}\sqrt{d_j}} \\ \frac{\partial R}{\partial u_i} &= (2+2\eta)u_i - 2\eta u_i^0 - 2\sum_{j=1}^{|P|}\frac{w_{ij}p_j}{\sqrt{d_i}\sqrt{d_j}}.\end{aligned} \tag{11}$$

Setting the derivatives to 0, we can obtain

$$\begin{aligned}p_j &= \frac{1}{1+\gamma}\sum_{i=1}^{|U|}\frac{w_{ij}}{\sqrt{d_i}\sqrt{d_j}}u_i + \frac{\gamma}{1+\gamma}p_j^0, \\ u_i &= \frac{1}{1+\eta}\sum_{j=1}^{|P|}\frac{w_{ij}}{\sqrt{d_i}\sqrt{d_j}}p_j + \frac{\eta}{1+\eta}u_i^0,\end{aligned} \tag{12}$$

which exactly recovers the BiRank iteration Eq. (2) by plugging $\gamma = \frac{1-\alpha}{\alpha}, \eta = \frac{1-\beta}{\beta}$ into the equation. As such, we see that BiRank is actually iterating towards optimizing the regularization function Eq. (10).

As we have shown BiRank converges to a stationary solution in Section 4.2.1. Now a question arises: does the solution found by BiRank lead to the regularization function's global optimum? In fact it does, as the regularization function is strictly convex in both p_j and u_i.

Theorem 3. *The regularization function $R(\mathbf{u}, \mathbf{p})$ defined by Eq. (10) is strictly convex and only one global minimum exists.*

Proof. According to convex optimization theory, a continuous, twice differentiable function is strictly convex if and only if its Hessian matrix is positive definite. As $R(\mathbf{u}, \mathbf{p})$ is a continuous function, we now prove it is twice differentiable and that its Hessian matrix is positive definite.

The second order derivative of $R(\mathbf{u}, \mathbf{p})$ is

$$\frac{\partial^2 R}{\partial p_j \partial p_j} = 2 + 2\gamma; \ \frac{\partial^2 R}{\partial u_i \partial u_i} = 2 + 2\eta; \ \frac{\partial^2 R}{\partial p_j \partial u_i} = 2\frac{-w_{ij}}{\sqrt{d_i}\sqrt{d_j}}.$$

We can see $R(\mathbf{u}, \mathbf{p})$ is twice differentiable.

Let matrix A be the $(|U| + |P|) \times (|U| + |P|)$ weighted adjacency matrix of the bipartite graph. Then the Hessian of $R(\mathbf{u}, \mathbf{p})$ can be written as: $2(I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) + 2B$, where D is a diagonal matrix where each entry D_{kk} denotes the weighted degree of kth vertex (can be of either side); B is a diagonal matrix that each entry B_{kk} is γ or η, dependant on the choice of origin (side) for the kth vertex.

Note that the matrix $(I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}})$ is the normalized Laplacian matrix of the graph. By spectral graph theory, the normalized Laplacian matrix of a graph is positive semi-definite. Meanwhile, B is also positive definite because its eigenvalues are all positive (eigenvalues of a diagonal matrix are its diagonal values). Finally, according to the standard linear algebra, the addition of a positive semi-definite matrix and positive definite matrix is also positive definite. Thus, we reach the conclusion that the Hessian matrix must be positive definite. The proof is finished. □

5.1.2 Interpretation of Regularization

It is instructive to interpret the meaning of the regularization function and see how it is constructed. First, it can be seen as enforcing two constraints in assigning the ranking scores on the vertices: A *smoothness* constraint that implies structural consistency—that nearby vertices should not vary much in their ranking scores; and a *fitting* constraint which encodes the query vector—that the ranking should not overly deviate from prior belief.

Smoothness. The first term of Eq. (10) implements the smoothness constraint, which constrains a vertex's normalized score to be similar to the normalized scores of its connected neighbors. Minimizing it leads to the simplified BiRank algorithm devised in Eq. (1). Moreover, it can be seen as the squared sum (L_2 distance) of *edge derivatives* on the graph, as introduced in graph regularization theory [18]

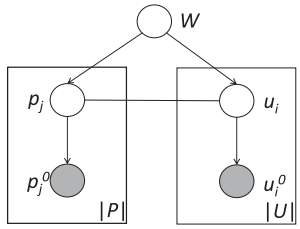


Fig. 2. Graphical model representation of BiRank.

$$\left.\frac{\partial f}{\partial e}\right|_{e_{ij}} = \sqrt{\frac{w_{ij}}{d_i}} u_i - \sqrt{\frac{w_{ij}}{d_j}} p_j,$$

which measures the variation (or energy drop) of the ranking function on edge e_{ij}. That is, if two vertices are strongly connected but exhibit a large difference in their scores, then the magnitude of the variation will be large. Variants of our vanilla BiRank can be derived by employing other methods to combine the edge derivatives, e.g., the L_1 distance, which can yield and model different effects for smoothness.

Fitting. The second and third term of the regularization function gives the fitting constraint for the query vectors $\mathbf{p}^0$ and $\mathbf{u}^0$, respectively

$$R_f(\mathbf{p}) = \sum_{j=1}^{|P|} (p_j - p_j^0)^2, \quad R_f(\mathbf{u}) = \sum_{i=1}^{|U|} (u_i - u_i^0)^2. \tag{13}$$

This fitting term is easy to understand: it regularizes the value of each vertex's score to be similar with its prior score, i.e., its value in the query vector.

In our formulation of BiRank, we have chosen a MSE (mean squared error) loss function form; other ranking-oriented loss functions, such as the BPR-OPT [28], may be more suited if one seeks to maintain the vertices' relative ordering in the query vector during the ranking process. We leave this possibility for future work.

5.2 Bayesian Explanation

On the basis of the above regularization framework, we now present a Bayesian explanation for BiRank.

Fig. 2 shows the graphical model representation of the ranking method. We model the query vectors $\mathbf{p}^0$ and $\mathbf{u}^0$ as observations, which are generated by the latent factors $\mathbf{p}$ and $\mathbf{u}$ (distributions), serving as the importance scores of the vertices; the weight matrix W forms the prior for generating the latent factors. The goal is to infer the latent factors $\mathbf{p}$ and $\mathbf{u}$ that generate the observations $\mathbf{p}^0$ and $\mathbf{u}^0$.

The MAP (maximum a posteriori) estimation is given by

$$\arg\max_{\mathbf{u},\mathbf{p}} p(\mathbf{u}, \mathbf{p} | \mathbf{u}^0, \mathbf{p}^0, W).$$

By Bayes' rule and the conditional independence indicated in the graphical model, we have

$$\begin{aligned} p(\mathbf{u}, \mathbf{p} | \mathbf{u}^0, \mathbf{p}^0, W) &= \frac{p(\mathbf{u}^0, \mathbf{p}^0 | \mathbf{u}, \mathbf{p}) \cdot p(\mathbf{u}, \mathbf{p} | W)}{p(\mathbf{u}^0, \mathbf{p}^0)} \\ &\propto p(\mathbf{u}^0, \mathbf{p}^0 | \mathbf{u}, \mathbf{p}) \cdot p(\mathbf{u}, \mathbf{p} | W) \\ &\propto p(\mathbf{u}^0 | \mathbf{u}) \cdot p(\mathbf{p}^0 | \mathbf{p}) \cdot p(\mathbf{u}, \mathbf{p} | W). \end{aligned}$$

Note that $\mathbf{p}$ and $\mathbf{u}$ are not conditionally independent with each other given the prior W, as a vertex's score is also influenced by its neighbors' scores. Taking the logarithm, MAP estimation is then equivalent to

$$\arg\max_{\mathbf{u},\mathbf{p}} \{\ln p(\mathbf{u},\mathbf{p}|W) + \ln p(\mathbf{u}^0|\mathbf{u}) + \ln p(\mathbf{p}^0|\mathbf{p})\}.$$

We then devise the conditional probabilities as follows:

$$p(\mathbf{u},\mathbf{p}|W) = \frac{1}{Z_{up}} e^{-R_s(\mathbf{u},\mathbf{p})},\ p(\mathbf{p}^0|\mathbf{p}) = \frac{1}{Z_p} e^{-\gamma R_f(\mathbf{p})},$$

$$p(\mathbf{u}^0|\mathbf{u}) = \frac{1}{Z_u} e^{-\eta R_f(\mathbf{u})},$$

where Z_{up}, Z_p and Z_u are normalization constants, and where $R_s(\mathbf{u},\mathbf{p})$ is the smoothness term, and $R_f(\mathbf{p})$ and $R_f(\mathbf{u})$ are the fitting terms defined in Eq. (13). From this formalization, we can see that minimizing the regularization function is equivalent to maximizing the posteriori probability of generating the query vector.

This shows the equivalence between BiRank's ranking process and a Bayesian network. We map the ranking problem to probabilistic graphical modeling, allowing the extension of BiRank in a probabilistic way, which is more flexible and adaptable for different applications. For example, if there is additional prior knowledge or context for the vertices, we can model them as priors of $\mathbf{p}, \mathbf{u}$ and use the desired distributions; moreover, aside from MAP, other inference techniques can also be applied to infer the ranking scores, such as variational inference and MCMC sampling.

5.3 Generalization to *n*-Partite Graph Ranking

Our proposed BiRank methodology is general and versatile. Here, we generalize it to rank vertices of the more general n-partite graphs. A n-partite graph is a graph whose vertices can be partitioned into n different independent sets. We represent it as $G(\{P_t\};\{E_{tl}\})$, where P represents vertices, E represents edges, t and l represent the indices of the independent vertex sets, satisfying $1 \le t, l \le n$. Let the weight matrix of edges E_{tl} be W_{tl}, which is a $|P_t| \times |P_l|$ matrix. If the graph is undirected, we have $W_{lt} = W_{tl}^T$. The symmetrically normalized matrix is defined as $S_{tl} = \sqrt{D^t} W_{tl} \sqrt{D^l}$, where D^t and D^l represent the diagonal degree matrix of vertices P_t and P_l, respectively. Let the ranking vector and query vector of vertices in P_t be $\mathbf{p}_t$ and $\mathbf{p}_t^0$, respectively. Then, the objective function for vertex ranking is defined by smoothing the connected vertices (of pairwise vertex types) and fitting the query vectors (of each vertex type)

$$R = \sum_t \eta_t ||\mathbf{p}_t - \mathbf{p}_0^t||^2 + \sum_{l \ne t} \gamma_{tl} \sum_{i,j} (W_{tl})_{ij} \left(\frac{(\mathbf{p}_t)_i}{\sqrt{D_{ii}^t}} - \frac{(\mathbf{p}_l)_j}{\sqrt{D_{jj}^l}} \right)^2,$$

where γ_{tl} and η_t are hyper-parameters that control the importance of the corresponding component. Similar to BiRank, this regularization function is strictly convex for all model parameters. Thereby, the global minimum can be achieved by alternating optimization, which leads to the iterative update solution as

$$\begin{aligned} \mathbf{p}_1 &= \sum_{l \ne 1} \alpha_{1l} S_{1l} \mathbf{p}_l + \left(1 - \sum_{l \ne 1} \alpha_{1l}\right) \mathbf{p}_1^0, \\ &\cdots\cdots \\ \mathbf{p}_t &= \sum_{l \ne t} \alpha_{tl} S_{tl} \mathbf{p}_l + \left(1 - \sum_{l \ne t} \alpha_{tl}\right) \mathbf{p}_t^0, \end{aligned} \tag{14}$$

where the hyper-parameters α_{tl} are associated with γ_{tl} and η_t, indicating the weight of graph substructure E_{tl} in contributing to the final ranking. Iteratively executing the above update rule until convergence, we obtain the ranking. We call this algorithm *n-partiteRank*, as a generalization of BiRank for n-partite graphs; it is easy to see when $n = 2$, the algorithm exactly recovers the BiRank. Also, the time complexity of the algorithm is linear to number of edges in the n-partite graph, which is very efficient for large-scale heterogeneous graphs in real-world applications. In Section 6.2, we demonstrate how to utilize this generic algorithm to model user reviews ($n = 3$, i.e., TriRank) for the application of personalized recommendation [29].

6 Applications

In this section, we demonstrate how to apply the BiRank method to two real-world applications; namely, 1) predicting the future popularity of items, and 2) recommending items of interest to users. We choose to model user comment data for addressing the relevant task, since it is a form of explicit feedback that is easily accessible to both content providers and external observers.[3]

6.1 Popularity Prediction

Predicting the popularity of web content has a wide range of applications, such as online marketing [30] and recommender system [26]. In what follows, we first briefly introduce the task, and then show how to customize BiRank to address the problem.

6.1.1 Task Introduction

A direct and objective metric to measure an item's popularity is the view count, which evaluates users' attention on the item. Thereby, previous works have primarily focused on modeling the view histories of items [30], [31] and casted the prediction as a regression task. However, for some external services (who are not the content providers themselves), items' view histories are not easily accessible. Specifically, most websites do not explicitly provide the view history for an item. Even in the cases where a website like YouTube and Flickr provides the current number of views, one will have to repeatedly and periodically crawl the item pages to build view histories, a very bandwidth intensive activity.

To assist external observers in predicting items' popularity, we are more interested in an alternative and more viable solution—modeling the affiliated user comments of items. In contrast to view count, the advantage of comments is the exposure of users' commenting activities up to the current time—crawling once, one can get the previous history and

3. In contrast, implicit feedback—such as users' clicks on webpages and views on items—is only obtainable for the internal content providers. For external observers, such as the third-party services, implicit feedback is usually difficult to access.

perform the prediction directly. However, the key deficiency is that the comment history can be much sparser than view history, since a user viewing an item may not comment on it. For example, it is common that two items have no comments during the time interval, while they attract views at a different rate. As such, existing view-based solutions [30], [31] (which are mostly regression-based methods) can fail to leverage the comment series to predict popularity accurately.

To tackle the sparsity issue in user comments for quality prediction, we need to account for more popularity signals in addition to the comment count. Here, we propose three ranking hypotheses observed from user comments that we wish to incorporate into our popularity prediction solution:

H1. Temporal Factor. If an item has received many comments recently, it is more likely to be popular in the near future. More recent comments are a salient signal that more users focused on the item recently.

H2. User Social Influence. If the users commenting on an item are more influential, the item is more likely to receive more views in the future. This is enabled by the Web 2.0 social interfaces that propagate a user's comments to friends and followers.

H3. Item Current Popularity. If an item has already been popular, it is likely to garner more views in the future. This is partially effected by the existing visual interfaces of Web 2.0 systems: the more views an item has, the more likely it will be promoted to users.

6.1.2 BiRank Customization

To customize BiRank for a certain ranking purpose, we need to construct the weighted bipartite graph model and devise the query vectors.

Bipartite Graph Construction. As we deal with users' commenting behaviors on items, we model their relationship as a bipartite graph-users and items form the two sides of vertices U and P, respectively, and edges E represent comments. If and only if a user has commented on an item, there is an edge between them. We use the edge weight to model the respective comment's contribution towards the item's future popularity. As the hypothesis $H1$ shows a strong near-term correlation, we assign w based on temporal considerations. Specifically, recent (older) comments should contribute more (less) to an item's future popularity. To achieve this, we choose a monotonically decreasing exponential decay function

$$w_{ij} = \delta^{a(t_0 - t_{ij}) + b}, \tag{15}$$

where δ is the decay parameter that controls the decay rate, t_0 is the ranking time and t_{ij} is the commenting time; a and b are constants, to be tuned for the particular media and site. Time units are arbitrary; they can be assigned as minutes, hours, days, weeks or other units, depending on the temporal resolution and the domain of the items to rank. If no edge exists between u_i and p_j, then w_{ij} is zero. In our empirical study, we find a setting of $\delta = 0.85, a = 1, b = 0$ leads to good performance, and thus use this setting across datasets. As we focus on short-term popularity prediction, we set the time unit as 1 day.

Query Vector Setup. We devise the user query vector $\mathbf{u}^0$ and item query vector $\mathbf{p}^0$ to account for the hypotheses $H2$ and $H3$, respectively. Intuitively, if a user has more friends, his behavior is likely to influence more users. Thus we set a user's prior score in the query vector proportional to the log value of his number of friends

$$u_i^0 = \frac{\log(1 + g_i)}{\sum_{k=1}^{|U|} \log(1 + g_k)},$$

where g_i is user u_i's number of friends at the ranking time. Note that we use add-1 smoothing to address the case where a user has no friends.

$H3$ models the current popularity factor on items. As such, the item query vector should encode our prior belief on each item's popularity prior to examining its recent comments. We capture this potential "rich-get-richer" effect by defining an item's score in the query vector as

$$p_j^0 = \frac{\log v_j}{\sum_{k=1}^{|P|} \log v_k},$$

where v_j denotes the view count of item p_j at ranking time.

After finalizing the edge weights and query vectors, the rationale in our design can be more clearly seen by looking into the BiRank iteration in Eq. (3). First, it captures the mutual reinforcement between users and items—the more recent the comments are by a user on an item, the higher the popularity score the item will receive; and in return, the popularity of the target item increases the user's influence. Second, the score of items and users is partially determined by the original setting of the query vector. To sum up, BiRank determines a user's social influence based on two source of evidence: his level of activity and his number of friends. Analogously, BiRank determines an item's future popularity based on four aspects: the frequency and recency of comments on it, the influence of the users commenting on it, and its current accumulated popularity. Thus, from a qualitative point of view, we see that the formulation of BiRank can encode our hypotheses on the ranking function.

6.2 Personalized Recommendation

In this section, we apply the generalized, n-partiteRank to the application of personalized recommendation. This is a more challenging task than popularity prediction, since it needs to generate a personalized ranking of items for each user. In what follows, we first show how to employ BiRank to encode the well-known collaborative filtering (CF) effect for recommendation. Then we use the TripartiteRank (short for TriRank) to additionally model aspects (extracted from comments' texts) for enhanced recommendation.

6.2.1 Collaborative Filtering with BiRank

In recommendation systems, collaborative filtering is the most successful and widely-used technique for personalization. It exploits user–item interactions (e.g., ratings, click histories) by assuming that users with similar interest consume similar items. The core of a CF algorithm lies in how it models the similarity among users and items. For example, neighbor-based CF [32] directly estimates the similarity by adopting statistical measure on the user–item matrix, while latent factor-based CF [26] estimates the similarity by projecting items/users into a latent space. Under our BiRank

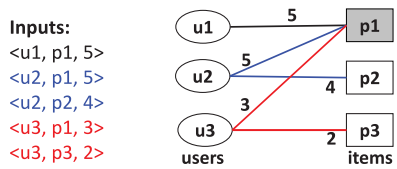


Fig. 3. A toy example of using BiRank to model the collaborative filtering effect. The target user u_1 has previously rated item p_1 with a rating score 5 (in tail).

TABLE 2
Top Automatically Extracted Aspects

Yelp	bar, salad, menu, chicken, sauce, restaurant, rice, cheese, fries, bread, sandwich, drinks
Amazon	camera, quality, sound, price, battery, pictures, screen, size, memory, lens

paradigm, similarity is estimated by means of smoothing the user–item graph with the target user's known preference, embodied as a query vector. We use an example in Fig. 3 to illustrate how the smoothness works.

Users and items represent the two types of vertices, and edge weights denote the rating scores (here, a zero score means the user did not rate the item; a missing value). Assume we want to recommend items to the target user u_1, who has rated p_1 with a score of 5. We construct the query vector by setting the prior of p_1 to 5 and other vertices to 0. Now, we consider how the BiRank predicts u_1's preference (i.e., the similarity to other items) with this setting. As p_1 is connected more strongly to u_2 than u_3, by the smoothness constraint, u_2 will be given a higher score than u_3. This indicates that BiRank treats u_2 more similar with u_1 than u_3. Finally, since the edge weights of $< u_2, p_2 >$ and $< u_3, p_3 >$ are identical, BiRank will assign p_2 a higher score than p_3, meaning that p_2 is a more suitable candidate to recommend for u_1 than p_3. From this qualitative analysis, we see that by properly setting the query vector's values, smoothing the user–item relation results in a collaborative filtering effect. More specifically, by setting the query vector as the rated items of the target user, BiRank functions similar to item-based CF [32] which represents a user by his historical actions for personalization.

6.2.2 Modelling Aspects with TriRank

Aside from ratings, which form the basis for collaborative filtering, most Web 2.0 systems also encourage users to pen reviews. These reviews justify a user's ratings, offering the underlying reasons for the rating by discussing the *specific properties* of the item. We term these specific properties as *aspects*, which are nouns or noun phrases that represent the features of items (see Table 2 as examples). Aspects are well-suited as a complementary data source for CF, since a mention of an aspect implies the user's interest in the aspect, which in turn reveals the user's preference. In this section, we model the aspects with TriRank to improve the CF-based recommendation. Similar to the application of BiRank to popularity prediction, we first show how to construct the tripartite graph, and then design the query vectors to implement the personalized ranking.

Tripartite Graph Construction. After extracting aspects from user reviews, we construct a tripartite graph with

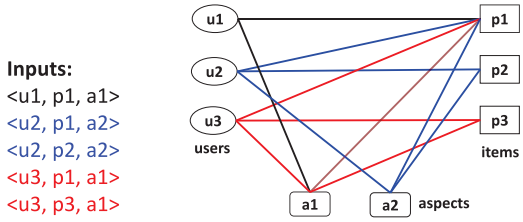


Fig. 4. An example of the tripartite graph (the dashed line illustrates the additional input $< u_1, p_3, a_2 >$).

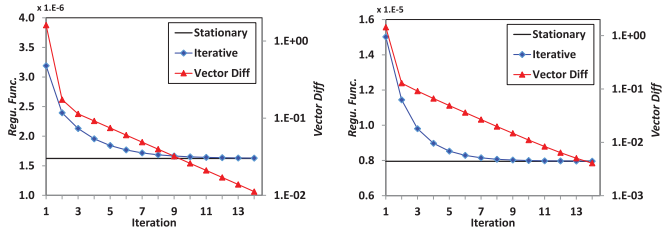
users, items and aspects as the three types of vertices. We formalize the input as a list of triples, where each triple $< u_i, p_j, a_k >$ denotes that user u_i has rated item p_j with a review mentioning aspect a_k and is represented as a triangle with edges e_{ij}, e_{ik} and e_{jk} in the graph. Fig. 4 shows an example of the tripartite graph.

Each edge carries a weight, which is crucial to determine the meaning of smoothness and the behavior of TriRank. The setting of *user-item* edge weights should encode the collaborative filtering effect: in cases with explicit feedback, it can be the rating score (as illustrated previously in Section 6.2.1); for implicit feedback, it can denote whether the user has interacted with or browsed the item (measured as either a binary yes/no, or an integer view count). Our datasets provide explicit user ratings, so we use these ratings as-is. The setting of aspect connected edges should reflect the aspect filtering effect: if a user is interested in an aspect, then the system should rank the items that are good at this aspect high. Thus, we set the edge weights of *user–aspect* relation and *item-aspect* relation to connote the degree of user interest (item specialty) with respect to the aspect. Once aspects are identified in reviews, we use the review frequency (i.e., number of reviews that mention the aspect) within all a user's (item's) reviews as the edge weight. As is done in general information retrieval, we take the logarithm of the review frequency, to dampen the effect of aspects that appear very frequently.

Query Vector Setup. The query vectors should encode the target user's prior preference on the vertices, which serve as the gateway for personalization. Here we discuss how to set the query vectors for target user u_i.

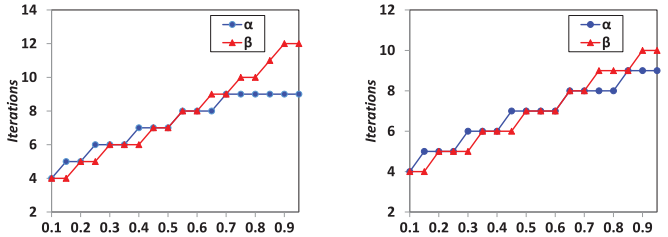
For the item query vector $\mathbf{p}_0$, an element takes a positive value if the target user has interacted with the item; otherwise, 0. Thus we adopt the ith row vector of the user–item matrix as the $\mathbf{p}_0$ for the target user u_i. Similarly, the aspect query vector $\mathbf{a}_0$ is set as the respective row vector of the user-aspect matrix, denoting the target user's prior preference on aspects. The user preference vector $\mathbf{u}_0$ should denote the target user's similarity with other users. When user's social network is available, we can use her friends information to initialize $\mathbf{u}_0$. Due to the lack of social information in our dataset, we adopt a simple approach, setting the target user herself as 1, and all other users as 0. Considering that the weight matrix is symmetrically normalized, we also apply the L_1 norm on $\mathbf{p}_0, \mathbf{a}_0$ and $\mathbf{u}_0$ respectively, for a meaningful combination.

Our final recommendation solution works as follows. After constructing the tripartite graph, we preform TriRank with the personalized query vector for each target user. The ranking process follows the iteration defined in Eq. (14).



(a) Random ($10K \times 50K$) (b) Power-law ($10K \times 50K$)

Fig. 5. Convergence status of two generated graphs.



(a) Rand (density=1%) (b) Power-law ($\lambda = 2$)

Fig. 6. Convergence rate *w.r.t.*α and β.

After convergence (usually in 10 iterations), items with the highest scores serve as the recommendations for the user, and the aspect vertices with the highest scores can be used as explanatory factors for the recommendations [29].

7 Experiments

In this section, we empirically examine BiRank's properties and effectiveness. We first conduct experiments on synthetic data to study BiRank's convergence and time efficiency. Then we perform experiments on real-world datasets to evaluate BiRank performance for the two applications of popularity prediction and personalized recommendation.

7.1 Experiments on Synthetic Data

7.1.1 Datasets

We concern ourselves with two forms of generated graphs:

1. Synthetic Random Graphs. These random graphs are generated by sampling edges from a uniform distribution. We control the density of generated bipartite graphs to simulate real-world graph sparsity. Given the expected density of the graph, we visit each potential edge and generate a uniformly random number in the range $(0, 1)$; if the number is less (or equal) than the density value, we add the edge into the graph.

2. Synthetic Power-Law Graphs. Considering that many real-world graphs follow a power-law distribution, such as document-word and user-item graphs, we also generate the power-law bipartite graphs. We adopt the power-law graph generation algorithm in [33]: starting from an empty graph, it follows two main steps: first it assigns a degree x to each vertex v from the distribution $p(d_v = x) \propto x^{-\lambda}$ where $\lambda > 1$; then, it sorts the vertices by degree in decreasing order, and assigns neighbors to each vertex according to the degree. We adjust the second step for generating bipartite graph—sampling neighbors of a vertex only from the vertices of the other side.

7.1.2 Convergence Study

There are two natural questions need to be answered empirically regarding BiRank's convergence[4]:

(1) Will BiRank iterations converge to the optimal solution of the regularization function as analyzed theoretically?
(2) How does the algorithm hyper-parameters (i.e., α and β) influence BiRank's convergence rate?

4. Note that due to the difficulty of controlling the eigenvalues of generated graphs, we do not empirically study the impact of the second dominant eigenvalue on convergence rate. While the impact has been theoretically proved in Theorem 2.

Since our findings were consistent across many settings, we only report results with a set of representative settings.

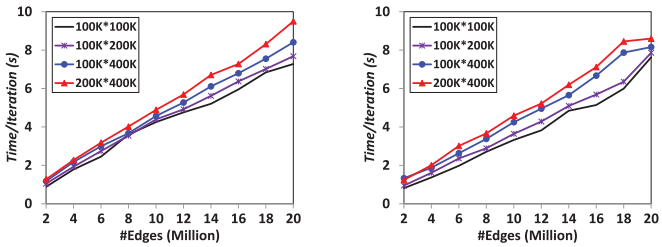
1. Convergence to Optimum. Fig. 5 plots the convergence status of each iteration on two representative synthetic graphs (the random setting has a density of 1 percent; the power-law graph sets $\lambda = 2$). The black line (Stationary) benchmarks the optimal solution from the direct calculation of the stationary solution Eq. (8). The blue line (Iterative) shows the regularization function's value after the update of each iteration; the red line (Vector Diff, y-axis scale of the right) shows the difference (i.e., squared sum) of the ranking vector before and after the update of each iteration. As we can see, BiRank successfully finds the optima of the regularization function in all four cases. Our further examination (not shown) validates that the ranking vector obtained by BiRank iterations is actually same with the stationary solution. This demonstrates BiRank's ability in converging to the unique and optimal solution of the regularization function, regardless of the graph structure. Moreover, the convergence rate is rather fast for these simulated problems—also usually within 10 iterations. Another finding is that the deepest descents are in the early iterations, which impose the most influence to the ranking.

2. Convergence Rate w.r.t. *Algorithm Parameters.* In BiRank, α and β are the hyper-parameters to combine the score calculated from the graph structure and query vector. They act like the damping factor in PageRank, and are crucial to the ranking results and convergence. We study the impact of the two parameters on the convergence rate. The convergence threshold is set as 0.0001 for Vector Diff, a strict condition that guarantees a sufficient convergence. Fig. 6 plots the number of iterations to converge on two graphs of size $10K \times 20K$. Both graphs show the same trend that BiRank needs more iterations to converge with a larger α (and β). This verifies our qualitative analysis in Section 4.2.2 that smaller value of α (and β) leads to a larger effect of the static query vectors, helpful in achieving quick convergence.

7.1.3 Time Efficiency

In Section 4.1.1, we analyzed the theoretic time complexity of BiRank: $O(n)$, in the number of graph edges. We now empirically validate this property. We adopt sparse representation for matrices, and implement BiRank in Java. The experiments are run on a modern desktop (Intel 3.5 GHz CPU with 16 GB RAM running on a single thread).

Fig. 7 shows the average time per iteration for graphs of different settings. First, from each single line, we see that the actual running time per iteration exhibits linearity *w.r.t.* to number of edges in the graph. More specifically, each



(a) Random Graphs (b) Power-law Graphs

Fig. 7. Running time per iteration *w.r.t.* number of edges.

TABLE 3
Demographics of the Three Web 2.0 Datasets

Dataset	Item#	User#	Comment#	Avg C/I	Crawled Date
YouTube	21,653	3,620,487	7,246,287	334.7	2012/8/9
Flickr	26,815	37,690	169,150	6.3	2012/9/3
Last.fm	16,284	77,996	530,237	32.6	2012/10/24

iteration takes about 0.9 seconds for graphs with 2 M edges, steadily increasing to 9 seconds for graphs with 20 M edges. This is rather efficient, given that we only run the algorithm in a single thread; for large-scale graphs, one can easily scale up the algorithm by parallelizing the matrix operations in multiple threads. Comparing across lines, we find that graphs of larger size take more time but still within the same magnitude. As the edge count is the same, the additional time is due to traversing additional vertices in such larger graphs when performing matrix operations.

7.2 Evaluation of Popularity Prediction

We now evaluate how does BiRank perform for the application of comment-based popularity prediction.

7.2.1 Experimental Settings

Datasets and Metrics. Table 3 shows the demographics of the three real-world datasets used in this evaluation. Each dataset is constructed by the search results of some seed queries. More details about the dataset are given in [1]. The evaluation ground-truth (GT) is the number of views (note, not the number of comments) received in the future three days after the original crawl date (i.e., ranking date t_0).

Given a set of items, BiRank outputs a ranking list of the items, indicating their predicted popularity. To assess the quality of the predicted ranking with the GT ranking globally, we adopt the Spearman coefficient, which measures the agreement between two rankings.

Baselines. We compare with the following six baselines:

1. *View Count (VC)*: Rank based on the current view count of items, corresponding to our Hypothesis *H3* alone.
2. *Comment Count in the Past (CCP)*: Rank based on the number of comments received in the 3-day period prior to t_0, corresponding to our Hypothesis *H1*.
3. *Multivariate Linear model (ML) [31]*: A state-of-the-art regression method for popularity prediction. We apply this method on the comment series with the time unit as 3 days. This baseline is to test the traditional view-based methods when applied to modeling comments.
4. *PageRank [2]*: This is the most widely used graph ranking method. Since the bipartite nature can cause the random walk to be non-stationary, we employ the standard method to set a uniform self-transition weight $w_{ii} = 1$ for all nodes before converting to a stochastic matrix.
5. *Co-HITS [8]*: This algorithm is devised for ranking on bipartite graphs by interleaving two random walks. To make a fair algorithmic comparison with BiRank, we apply the same query vectors to Co-HITS and tune the parameters in the same way.
6. *BGER [20]*: This is another algorithm designed for ranking on bipartite graphs. Instead of simulating a random walk, it normalizes the edge weights in a different way and is analogously explained as heat diffusion. We apply the same query vectors and parameter search for this method.

To expedite parameter tuning, we randomly held out 10 percent of the dataset as the development set, and employ grid search to find the optimal parameters. Then the performance is evaluated on the remaining 90 percent as the testing items.

TABLE 4
Popularity Prediction Evaluated by Spearman Coefficient (%)

Method	YouTube	Flickr	Last.fm
1. VC	73.39	58.42	67.31
2. CCP	83.35	59.43	67.21
3. ML	78.24	58.00	38.09
4. PageRank	80.72	28.15	10.24
5. Co-HITS	85.21	63.81	**72.71***
6. BGER	84.10	63.17	68.94
7. BiRank (ours)	**88.21***	**64.76***	70.93

"" denotes the statistical significance for $p < 0.01$ judged by the one-sample paired t-test.*

7.2.2 Performance Comparison

Table 4 shows the performance of the methods on the three datasets. First, we can see that the three bipartite graph ranking methods (lines 5, 6, and 7) significantly outperform other methods. This is because these methods model all the three ranking hypotheses we proposed, while other methods only partially model the hypotheses. Among the three bipartite ranking methods, BiRank achieves the best performance in general (best on two datasets YouTube and Flickr), followed by Co-HITS (best on Last.fm) and BGER. Further experimentation of 10-fold cross validation shows that the improvements of BiRank over Co-HITS and BGER on YouTube and Flickr datasets are consistent and statistically significant ($p < 0.01$, via one-sample paired t-test). Moreover, Co-HITS outperforms BGER consistently, although the random walk treatments of Co-HITS are suspicious to bias the high-degree vertices while BGER does not have this issue. We suspect the reason of Co-HITS's strong performance might be that the bias effect is diluted by the setting of query vectors, which can regulate the random walks effectively.

Focusing on the result of PageRank (line 4), we see that it performs very poorly for Flickr and Last.fm. This indicates

TABLE 5
Statistics of Datasets in Evaluation

Dataset	Review#	Item#	User#	Aspect#
Yelp	114,316	4,043	3,835	6,025
Amazon	55,677	14,370	2,933	1,617

that just the centrality of an item in the user–item temporal graph is insufficient for accurate popularity prediction. In addition, the performance discrepancy between PageRank and CoHITS (also a random walk-based method) highlights the importance of separately handling the two vertex types within the bipartite graph.

It is surprising that the regression approach ML underperforms CCP, as ML leverages more information: comments in the recent 30 days compared with CCP's access to only three days. We believe there are two reasons for this: 1) the nature of short-term prediction, and 2) the sparsity of comments. As the prediction task is a short-term one, the most recent data carries the most signal—"What happened yesterday will happen tomorrow"; the performance score of CCP verifies this point. Second, the sparsity in comment series (e.g., some time units have zero count) can negatively affect the regression process in an unexpected manner.

7.3 Evaluation of Personalized Recommendation

In this section, we study how do our BiRank and TriRank perform for the task of personalized item recommendation.

7.3.1 Experimental Settings

Datasets. We experiment with two public review datasets: Yelp[5] and Amazon Electronics.[6] We follow the common practice in evaluating recommendation algorithms [26], [28] that filters out users and items with fewer than 10 reviews. We used the sentiment analysis tool developed by [34] for extracting aspects from review texts. Table 5 summarizes the statistics of the filtered datasets and Table 2 shows examples of the top aspects extracted.

Baselines. We compare with the following methods that are commonly used in top-K recommendation:

1. *Popularity (ItemPop).* Items are ranked by their popularity judged by number of ratings. This non-personalized method benchmarks the performance of the top-K task.
2. *ItemKNN [32].* This is standard item-based CF. We tested the method with different number of neighbors, finding that using all neighbors works best.
3. *PureSVD [35].* A state-of-the-art model-based CF method for top-K recommendation, which performs SVD on the whole matrix. We tuned the number of latent factors.
4. *PageRank [4].* This graph method has been widely used for top-K recommendation, such as by [36]. For a fair comparison, we set the personalized vector the same with TriRank's query vectors and tuned the damping factor.

5. yelp.com/dataset_challenge. Downloaded on October 2014.
6. snap.stanford.edu/data/web-Amazon-links.html

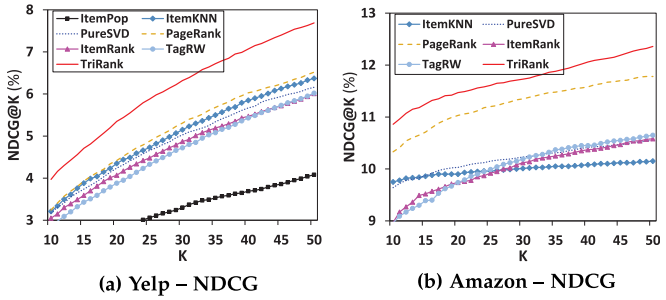


Fig. 8. Performance comparison of top-K recommendation evaluated by NDCG from position 10 to 50 (i.e., K).

TABLE 6
Recommendation Performance (%) Evaluated at Rank 50

Dataset	Yelp		Amazon	
Metric(%)	HR	NDCG	HR	NDCG
1. ItemPop	10.61	4.08	6.13	2.37
2. ItemKNN	15.72	6.37	12.69	10.15
3. PureSVD	14.94	6.16	14.94	10.55
4. BiRank (ours)	**17.00***	**6.90***	**15.97***	**11.16***
5. PageRank	15.90	6.52	17.49	11.78
6. TagRW	15.25	6.02	17.47	10.65
7. TriRank (ours)	**18.58***	**7.69***	**18.44***	**12.36***

BiRank outperforms CF-based methods (Lines 1, 2, and 3) and TriRank outperforms all other methods.

5. *TagRW [37].* A state-of-the-art tag-based recommendation solution, which performs random walks on the user–user and item–item similarity graph. Since tags have a similar form with aspects, we feed aspect as tags into the method.

For each user, we sort her reviews in chronological order. The first 80 percent are used for training, followed by 10 percent as validation (for parameter tuning) and 10 percent as test set (for evaluation). Given a test user, we assess the ranked list of top-K items with *Hit Ratio* [28] and *NDCG* [29].

7.3.2 Performance Comparison

Fig. 8 plots the performance of top-K recommendation methods evaluated by NDCG from position 10 to 50. Performance of hit ratio shows a similar trend with NDCG, and is thus omitted for space. ItemPop performed very weakly on the Amazon dataset, and is entirely omitted in Fig. 8b to better highlight the performance of the other methods. As can be seen, our TriRank consistently outperforms the baselines with a large margin, and the one-sample paired t-test verifies that the improvements over all baselines are statistically significant with $p < 0.01$. For a more detailed discussion, we further show the concrete scores obtained at the position 50 in Table 6.

Focusing on Lines 1, 2, 3, and 4 that are all CF methods that only model the user–item relationship, we see that our BiRank achieves the best performance on both datasets; specifically, it improves over the competitive recommendation methods ItemKNN and PureSVD with a relative improvement about 8.3 percent. This is very encouraging, and gives evidence of the merit of our specification of BiRank (in Section 6.2.1) for collaborative filtering. ItemKNN performs

very well on the Yelp dataset (better than PureSVD), but poorly on the Amazon one. One possible reason comes from data sparsity: as in Table 5, each item of the Amazon dataset only has 3.9 reviews on average. In such cases, the statistical similarity measure may fail in neighbor-based CF. In contrast, model-based methods are more robust to sparse data by projecting users and items to the latent space. Lastly, we see Item Popularity performs the worst, indicating the importance of modeling users personalized preferences, rather than just recommending popular items.

Moving to Lines 5, 6, and 7 of review-based methods, we see that they generally improve over the methods that use CF only, indicating the utility of reviews (more specifically, item aspects) for uncovering users' preference and complementing with user ratings. Second, TriRank achieves the best performance, further improving over BiRank with over a 10 percent relative improvement and outperforming PageRank and TagRW significantly. This verifies the effectiveness of our TriRank in incorporating the aspects for enhanced recommendation. Lastly, TagRW is inferior to PageRank in utilizing the same aspect source. We believe the main reason comes from TagRW's transformation of the user-item-aspect graph to user-user and item-item graphs, which can cause some signal loss especially when the original relationships are sparse.

8 Conclusion

We focus on the problem of ranking vertices of bipartite graphs, and more generally, n-partite graphs. We devise a new, generic algorithm—BiRank—which ranks vertices by accounting for both the graph structure and prior knowledge. BiRank is theoretically guaranteed to converge to a stationary solution, and can be explained from both a regularization view and a Bayesian view. This appealing feature allows future extensions to BiRank to be grounded in a principled way. To demonstrate the efficacy of our proposal, we examine two ranking scenarios: a general ranking scenario of item popularity prediction by modeling the user-item binary relationship, and a personalized ranking scenario of item recommendation by modeling the user-item-aspect ternary relationship. By properly setting the graph's edge weights and query vectors, BiRank can be customized to encode various ranking hypotheses. Extensive experiments on both synthetic and real datasets demonstrate the effectiveness of our method. In future, we will study how to optimally learn the hyper-parameters of BiRank. Owing to the two views of BiRank, two solutions can be explored—by adapting the parameters based on the validation set [38], or by integrating over the parameters under the Bayesian network formalism. Moreover, we will explore how to integrate the graph regularization framework with matrix factorization methods, which have been shown to be very effective for many tasks such as recommendation [26] and clustering [39].

Acknowledgments

This research is supported by the National Key Research and Development Program of China (No. 2016YFB1000905) and NSFC under Grant No. U1401256. NExT research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative. The authors thank the anonymous reviewers for their valuable comments, and acknowledge the additional discussion and help from Liqiang Nie, Jun-Ping Ng, Tao Chen and Yiqun Liu. This paper is an extended version of the SIGIR'14 conference paper [1]. Xiangnan He is the corresponding author.

References

[1] X. He, M. Gao, M.-Y. Kan, Y. Liu, and K. Sugiyama, "Predicting the popularity of web 2.0 items based on user comments," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 233–242.

[2] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Stanford InfoLab, Stanford, CA, USA, Tech. Rep. 1999–66, 1999.

[3] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[4] T. H. Haveliwala, "Topic-sensitive PageRank," in *Proc. 11th Int. Conf. World Wide Web*, 2002, pp. 517–526.

[5] R. Lempel and S. Moran, "The stochastic approach for link-structure analysis (SALSA) and the TKC effect," *Comput. Netw.*, vol. 33, no. 1, pp. 387–401, 2000.

[6] Y. Liu, et al., "BrowseRank: Letting web users vote for page importance," in *Proc. 31st Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2008, pp. 451–458.

[7] B. Gao, T.-Y. Liu, W. Wei, T. Wang, and H. Li, "Semi-supervised ranking on very large graphs with rich metadata," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 96–104.

[8] H. Deng, M. R. Lyu, and I. King, "A generalized Co-HITS algorithm and its application to bipartite graphs," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 239–248.

[9] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos, "Relevance search and anomaly detection in bipartite graphs," *ACM SIGKDD Explorations Newslett.*, vol. 7, no. 2, pp. 48–55, 2005.

[10] X. Li, M. Zhang, Y. Liu, S. Ma, Y. Jin, and L. Ru, "Search engine click spam detection based on bipartite graph propagation," in *Proc. 7th ACM Int. Conf. Web Search Data Mining*, 2014, pp. 93–102.

[11] L. Getoor and C. P. Diehl, "Link mining: A survey," *ACM SIGKDD Explorations Newslett.*, vol. 7, no. 2, pp. 3–12, 2005.

[12] A. Y. Ng, A. X. Zheng, and M. I. Jordan, "Stable algorithms for link analysis," in *Proc. 24th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2001, pp. 258–266.

[13] C. Ding, X. He, P. Husbands, H. Zha, and H. D. Simon, "PageRank, hits and a unified framework for link analysis," in *Proc. 25th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2002, pp. 353–354.

[14] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 321–328.

[15] A. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning Theory and Kernel Machines*. Berlin Germany: Springer, 2003, pp. 144–158.

[16] S. Agarwal, "Ranking on graph data," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 25–32.

[17] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schlkopf, "Ranking on data manifolds," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 169–176.

[18] D. Zhou and B. Schölkopf, "Regularization on discrete spaces," in *Pattern Recognition*. Berlin, Germany: Springer, 2005, pp. 361–368.

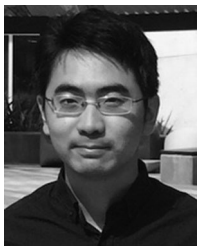
[19] D. Zhou, J. Huang, and B. Schölkopf, "Learning from labeled and unlabeled data on a directed graph," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 1036–1043.

[20] L. Cao, J. Guo, and X. Cheng, "Bipartite graph based entity ranking for related entity finding," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.*, 2011, pp. 130–137.

[21] X. Rui, M. Li, Z. Li, W.-Y. Ma, and N. Yu, "Bipartite graph reinforcement model for web image annotation," in *Proc. 15th ACM Int. Conf. Multimedia*, 2007, pp. 585–594.

[22] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh, "WTF: The who to follow service at Twitter," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 505–514.

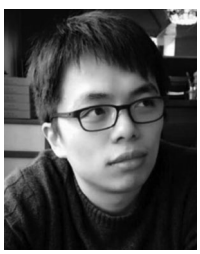
[23] S. Baluja, et al., "Video suggestion and discovery for YouTube: Taking random walks through the view graph," in *Proc. 17th Int. Conf. World Wide Web*, 2008, pp. 895–904.
[24] D. Parveen and M. Strube, "Multi-document summarization using bipartite graphs," in *Proc. TextGraphs-9 Workshop Graph-Based Methods Natural Language Process.*, 2014, pp. 15–24.
[25] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Baltimore, MD, USA: Johns Hopkins Univ. Press, vol. 3, 2012.
[26] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2016, pp. 549–558.
[27] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua, "Discrete collaborative filtering," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2016, pp. 325–334.
[28] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, 2009, pp. 452–461.
[29] X. He, T. Chen, M.-Y. Kan, and X. Chen, "TriRank: Review-aware explainable recommendation by modeling aspects," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 1661–1670.
[30] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," *Commun. ACM*, vol. 53, pp. 80–88, 2010.
[31] H. Pinto, J. M. Almeida, and M. A. Gonçalves, "Using early view patterns to predict the popularity of YouTube videos," in *Proc. 6th ACM Int. Conf. Web Search Data Mining*, 2013, pp. 365–374.
[32] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 285–295.
[33] M. Gao, E.-P. Lim, and D. Lo, "R-energy for evaluating robustness of dynamic networks," in *Proc. 5th Annu. ACM Web Sci. Conf.*, 2013, pp. 89–98.
[34] Y. Zhang, H. Zhang, M. Zhang, Y. Liu, and S. Ma, "Do users rate or review?: Boost phrase-level sentiment labeling with review-level sentiment classification," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 1027–1030.
[35] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-N recommendation tasks," in *Proc. 4th ACM Conf. Recommender Syst.*, 2010, pp. 39–46.
[36] S. Lee, S.-I. Song, M. Kahng, D. Lee, and S.-G. Lee, "Random walk based entity ranking on graph for multidimensional recommendation," in *Proc. 5th ACM Conf. Recommender Syst.*, 2011, pp. 93–100.
[37] Z. Zhang, D. D. Zeng, A. Abbasi, J. Peng, and X. Zheng, "A random walk model for item recommendation in social tagging systems," *ACM Trans. Manage. Inf. Syst.*, vol. 4, no. 2, pp. 1–24, 2013.
[38] S. Rendle, "Learning recommender systems with adaptive regularization," in *Proc. 5th ACM Int. Conf. Web Search Data Mining*, 2012, pp. 133–142.
[39] X. He, M.-Y. Kan, P. Xie, and X. Chen, "Comment-based multi-view clustering of web 2.0 items," in *Proc. 23rd Int. Conf. World Wide Web*, 2014, pp. 771–782.



Xiangnan He is currently a postdoctoral research fellow in the School of Computing, National University of Singapore. His research interests include information retrieval, recommender systems, multimedia, and machine learning. His works have appeared in several major international conferences such as SIGIR, WWW, MM, CIKM, and AAAI. He has served as a program committee member of international conferences such as SIGIR, WWW, and EMNLP.

Ming Gao received the doctorate degree from the School of Computer Science, Fudan University. He is an associate professor in the Software Engineering Institute, East China Normal University, China. His research interests include uncertain data management, streaming data processing, social network analysis, and data mining. His work appears in major international conferences including ICDE, ICDM, DASFAA, and WebSci. He is a member of the IEEE.

Min-Yen Kan is an associate professor with the National University of Singapore. He previously served as a member of the executive committee of the Association of Computational Linguistics (ACL) and maintains the ACL Anthology, the community's largest archive of published research. He is an associate editor of the Springer *Information Retrieval* journal. His research interests include digital libraries and applied natural language processing and information retrieval. Specific projects include work in the areas of scientific discourse analysis, full-text literature mining, machine translation, lexical semantics, and applied text summarization. He is a member of the IEEE.



Dingxian Wang is a research engineer in the Search Science Department, eBay. He is specifically working on the project of product classification for improving search ranking. His research interests include information retrieval, software engineering, natural language processing, and applied machine learning. His works have appeared in international conferences including WISE, CSE, and ICSSP.