Learning Vertex Representations for Bipartite Networks

Ming Gao, Xiangnan He, Leihui Chen, and Aoying Zhou

Abstract—Recent years have witnessed a widespread increase of interest in network representation learning (NRL). By far most research efforts have focused on NRL for homogeneous networks like social networks where vertices are of the same type, or heterogeneous networks like knowledge graphs where vertices (and/or edges) are of different types. There has been relatively little research dedicated to NRL for bipartite networks. Arguably, generic network embedding methods like node2vec and LINE can also be applied to learn vertex embeddings for bipartite networks by ignoring the vertex type information. However, these methods are suboptimal in doing so, since real-world bipartite networks concern the relationship between two types of entities, which usually exhibit different properties and patterns from other types of network data. For example, E-Commerce recommender systems need to capture the collaborative filtering patterns between customers and products, and search engines need to consider the matching signals between queries and webpages.

This work addresses the research gap of learning vertex representations for bipartite networks. We present a new solution BiNE, short for ***Bi**partite **N**etwork **E**mbedding*, which accounts for two special properties of bipartite networks: long-tail distribution of vertex degrees and implicit connectivity relations between vertices of the same type. Technically speaking, we make three contributions: (1) We design a biased random walk generator to generate vertex sequences that preserve the long-tail distribution of vertices; (2) We propose a new optimization framework by simultaneously modeling the explicit relations (*i.e.,* observed links) and implicit relations (*i.e.,* unobserved but transitive links); (3) We explore the theoretical foundations of BiNE to shed light on how it works, proving that BiNE can be interpreted as factorizing multiple matrices. We perform extensive experiments on five real datasets covering the tasks of link prediction (classification) and recommendation (ranking), empirically verifying the effectiveness and rationality of BiNE. Our experiment codes are available at: https://github.com/clhchtcjj/BiNE.

Index Terms—Bipartite networks, Network representation learning, Matrix factorization, Link prediction, Recommendation

1 Introduction

Network provides a ubiquitous data structure to model interactions (*i.e.,* edges) among entities (*i.e.,* vertices), having been widely used in many applications such as social networks [2], knowledge graphs [3], recommender systems [4], among others [5]. However, performing predictive analytics or mining knowledge directly on networks exhibits several challenges, such as high computation complexity, low parallelizability, and inapplicability of machine learning methods [6]. To handle these challenges, substantial works focused on network representation learning (NRL). In particular, they represent a vertex as a learnable embedding vector, where the proximity between vectors encodes the information about network structure. Thus the vertex embedding can be fed into machines learning methods to address various task such as ranking, link prediction, clustering, visualization and so on.

Recent advances in NRL have primarily focused on homogeneous network like social networks where vertices having the same type [7], [8], [9], [10], or heterogeneous networks like knowledge graphs where vertices (and/or edges) are of different types [11], [12], [13]. The effectiveness and prevalence of DeepWalk [8] inspire many works [8], [14] typically apply a two-step solution: 1) performing random walks, such as based random walk [14] or meta-path-based random walk [11], on the network to obtain a "corpus" of vertices; 2) applying embedding methods on the corpus such as word2vec [15] to obtain the embeddings for vertices.

However, there has been relatively little research dedicated to NRL for bipartite networks, for which there are two types of vertices [16]. While generic network embedding methods like DeepWalk and node2vec can be applied to learn vertex representation for bipartite networks by ignoring the the vertex type information, we argue that these methods are suboptimal in doing so because of two reasons. (1) Real-world bipartite networks concern the relationship between two types of entities, which usually exhibit different properties and patterns from other types of network data. For example, E-Commerce recommender system models the collaborative filtering patterns between customers and products, and Web search engine considers the matching signals between queries and webpages. (2) The generated corpus may not preserve the characteristics of the bipartite network. The power-law distribution is a common characteristic of many real-world bipartite networks [17], but the corpus generated by a universal random walk algorithm may not preserve this property, such as the one used in DeepWalk [8]. Specifically, it generates the same number of random walks starting from each vertices and constrains the length of random walks to be the same, which may

This paper is an extended version of the SIGIR '18 conference paper [1]. Xiangnan He is the corresponding author.

- *Ming Gao, Leihui Chen, and Aoying Zhou are with the School of Data Science and Engineering, East China Normal University, Shanghai, China, 200062. {mgao, ayzhou}@dase.ecnu.edu.cn, leihuichen@gmail.com*
- *Xiangnan He is with the School of Computing, National University of Singapore, Singapore. xiangnanhe@gmail.com.*

limit the the information of vertices with high degree and oversample vertices with low degree.

To address the limitations of exiting methods on embedding bipartite networks, we devise a new solution for learning vertex representation in bipartite networks, namely BiNE (short for ***Bi**partite **N**etwork **E**mbedding*). We summarize the main contributions of this work as follows.

1) We propose a biased and self-adaptive random walk generator to preserve the long-tail distribution of vertex in bipartite networks as much as possible. Specifically, we set the number of random walks starting from each vertex based on its importance and allow each walk to be stopped in a probabilistic way instead of setting a uniform length for all random walks.
2) We propose a joint optimization framework to model the explicit and implicit relations simultaneously. Specifically, we devise a dedicated objective function for each relation and optimize them by sharing the embedding vectors, where different relations reinforce each other and lead to better vertex embeddings.
3) We reveal the theoretical foundation of BiNE, which can be interpreted as implicitly factorizing multiple matrices, offering a better understanding how BiNE works.
4) We perform extensive experiments on several real datasets covering the tasks of links prediction (classification), recommendation (personalized ranking), and visualization, to illustrate the effectiveness and rationality of BiNE.

A preliminary version of this work has been published in the conference of SIGIR 2018 [1]. We summarize the main changes as follows:

1) Introduction (Section 1). We have reconstructed the abstract and introduction to highlight the motivations of the extended version.
2) Theoretical foundations (Section 4). We explore the theoretical foundations of BiNE to shed light on how it works, proving that BiNE can be interpreted as factorizing multiple matrices.
3) Experiments (Section 5). We add experiments to verify the proof, showing that the factorization-based implementation can achieve the same level of performance as BiNE in Section 5.3. In addition, we explore the performance of BiNE with different negative sampling strategies to justify our sampler design in Section 5.5.

The remainder of the paper is organized as follows. We first formulate the problem in Section 2, before delving into details of the method in Section 3. We show the theoretical connections with factorization methods in Section 4 and perform empirical studies in Section 5. We review related work in Section 6 before concluding the paper in Section 7.

2 Problem Formulation

We first give notations used in this paper, and then formalize the bipartite network embedding problem to be addressed.

Notations. Let $G = (U, V, E)$ be a bipartite network, where U and V denote the set of the two types of vertices respectively, and $E \subseteq U \times V$ defines the inter-set edges. u_i and v_j denote the i-th and j-th vertex in U and V, respectively, where $i = 1, 2, ..., |U|$ and $j = 1, 2, ..., |V|$. Each edge carries a non-negative weight w_{ij}, describing the strength between the connected vertices u_i and v_j; if u_i and v_j are not connected, the edge weight w_{ij} is set to zero. Therefore, we can use a $|U| \times |V|$ matrix $\mathbf{W} = [w_{ij}]$ to represent the weighted structure of the bipartite network.

Problem Definition. The aim of bipartite network embedding is to map all vertices in the network into a low-dimensional embedding space, where each vertex is represented as a dense vector. In the embedding space, both the implicit relations between vertices of the same type and the explicit relations between vertices of different types should be preserved. Formally, the problem can be defined as:

Input: A bipartite network $G = (U, V, E)$ and its weight matrix $\mathbf{W}$.

Output: A mapping function $f : U \cup V \rightarrow \mathbb{R}^d$, which maps each vertex in G to a d-dimensional embedding vector.

To keep the notations simple, we use $\vec{\mathbf{u}}_i$ and $\vec{\mathbf{v}}_j$ to denote the embedding vectors for vertices u_i and v_j, respectively. As such, we can present the embedding vectors of all vertices in the bipartite network as two matrices $\mathbf{U} = [\vec{\mathbf{u}}_i]$ and $\mathbf{V} = [\vec{\mathbf{v}}_j]$.

3 BiNE: Bipartite Network Embedding

The typical objective in learning vertex embeddings is to be capable of reconstructing the network structure well [8], [14]. While the structure of normal networks is mostly reflected in the observed edges, the case is more complicated for bipartite networks — two vertices of the same type are not directly connected, but it doesn't necessarily mean that they do not have relation. This poses challenges to bipartite network embedding, such that modeling only observed edges is insufficient to retain the fidelity. Towards this end, we propose to account for both the observed edges (Section 3.1 Modeling Explicit Relations) and the unobserved but transitive edges (Section 3.1 Modeling Implicit Relations). The final vertex embeddings are achieved by jointly optimizing the two tasks (Secion 3.3 Joint Optimization).

3.1 Modeling Explicit Relations

Edges between vertices of different types in a bipartite afford us a signal to capture the explicit structure information. Similar to the modeling of 1st-order proximity in LINE [18], we preserve the explicit structure information by minimizing the difference between the empirical distribution of vertex co-occurring probability and the reconstructed distribution by the vertex embeddings. The co-occurring probability between two connected vertices u_i and v_j in the original bipartite network is defined as:

$$P(i, j) = \frac{w_{ij}}{\sum_{e_{st} \in E} w_{st}}. \tag{1}$$

where w_{ij} is the weight of edge e_{ij}. In addition, the local proximity between them in the embedding space can be estimated by their inner product [8], [14], [18], we further transform this interaction value to the probability space by the sigmoid function:

$$\hat{P}(i, j) = \frac{1}{1 + exp(-\vec{\mathbf{u}}_i^T \vec{\mathbf{v}}_j)}. \tag{2}$$

where $\vec{\mathbf{u}}_i \in \mathbb{R}^d$ and $\vec{\mathbf{v}}_j \in \mathbb{R}^d$ are the embedding vectors of vertices u_i and v_j, respectively.

After getting the empirical distribution and the reconstructed distribution, we employ the KL-divergence to measure the difference between the two distributions, and learn

the embedding vectors by minimizing the difference. Thus the objective function can be defined as:

$$\begin{aligned} minimize \quad O_1 &= KL(P||\hat{P}) = \sum_{e_{ij}\in E} P(i,j)\log(\frac{P(i,j)}{\hat{P}(i,j)}) \\ &\propto -\sum_{e_{ij}\in E} w_{ij}\log\hat{P}(i,j). \end{aligned} \tag{3}$$

Intuitively, two strongly connected vertices in the original network will be close with each other in the embedding space by minimizing the objective function. Thus the explicit structure information can be preserved.

3.2 Modeling Implicit Relations

The effectiveness of modeling implicit relations in recommendation [19], [20] (which deals with user-item bipartite network) motivates us to explore the implicit relations in bipartite networks towards real-world applications. Although a perfect reconstruction of explicit relations can fully recover the implicit relations, it is impractical to rely on this. As such, we speculate that modeling the implicit relations between vertices of the same type could bring extra benefits to explicit relation modeling. Intuitively, if there exists a path between two vertices, it implies certain implicit relation between them; the number of the paths and their length indicate the strength of the relation. However, counting the paths between two vertices comes at the cost of very high complexity, which is unaffordable for large-scale networks. To encode such high-order implicit relations among vertices in a bipartite network, we resort to the solution of DeepWalk. To be exact, the bipartite network is first converted to two corpora of vertex sequences by performing random walks; then the embeddings are learned from the corpora which encodes high-order relations between vertices.

3.2.1 *Constructing Corpus of Vertex Sequences*

It is a common way to convert a network into a corpus of vertex sequences by performing random walks on the network, which has been used in some homogeneous network embedding methods [8], [14]. However, directly performing random walks on a bipartite network could fail, since there is no stationary distribution of random walks on bipartite networks due to the periodicity issue [21]. To address this issue, we consider performing random walks on two homogeneous networks that contain the 2nd-order proximity between vertices of the same type. Following the idea of Co-HITS [22], we define the 2nd-order proximity between two vertices as:

$$w_{ij}^U = \sum_{k\in V} w_{ik}w_{jk}; \quad w_{ij}^V = \sum_{k\in U} w_{ki}w_{kj}. \tag{4}$$

where w_{ij} is the weight of edge e_{ij}. Hence, we can use the $|U|\times|U|$ matrix $\mathbf{W}^U = [w_{ij}^U]$ and the $|V|\times|V|$ matrix $\mathbf{W}^V = [w_{ij}^V]$ to represent the two induced homogeneous networks, respectively.

Now we can perform truncated random walks on the two homogeneous networks to generate two corpora for learning the high-order implicit relations. To generate a corpus with a high fidelity, we propose a biased and self-adaptive random walk generator, which can preserve the vertex distribution in a bipartite network. We highlight its core designs as follows:

- First, we relate the number of random walks starting from each vertex to be dependent on its importance, which can be measured by its centrality. For a vertex, the greater its centrality is, the more likely a random walk will start from it. As a result, the vertex importance can be preserved to some extent.
- We assign a probability to stop a random walk in each step. In contrast to DeepWalk and other work [11] that apply a fixed length on the random walk, we allow the generated vertex sequences have a variable length, in order to have a close analogy to the variable-length sentences in natural languages.

Generally speaking, the above generation process follows the principle of "rich gets richer", which is a physical phenomena existing in many real networks, i.e., the vertex connectivities follow a scale-free power-law distribution [23].

The workflow of our random walk generator is summarized in Algorithm 1, where $maxT$ and $minT$ are the maximal and minimal numbers of random walks starting from each vertex, respectively. $\mathcal{D}^U$ (or $\mathcal{D}^V$) output by Algorithm 1 is the corpus generated from the vertex set U (or V). The vertex centrality can be measured by many metrics, such as degree centrality, PageRank and HITS [24], etc., and we use HITS in our experiments.

Algorithm 1: WalkGenerator(W, R, $maxT$, $minT$, p)

Input : weight matrix of the bipartite network **W**, vertex set R (can be U or V), maximal walks per vertex $maxT$, minimal walks per vertex $minT$, walk stopping probability p

Output: a set of vertex sequences $\mathcal{D}^R$

```
Calculate vertices' centrality: H = CentralityMeasure(W);
Calculate W^R w.r.t. Equation (4);
foreach vertex v_i ∈ R do
    l = max(H(v_i) × maxT, minT);
    for i = 0 to l do
        D_{v_i} = BiasedRamdomWalk(W^R, v_i, p);
        Add D_{v_i} into D^R;
return D^R;
```

3.2.2 *Implicit Relation Modeling*

After performing biased random walks on the two homogeneous networks respectively, we obtain two corpora of vertex sequences. Next we employ the Skip-gram model [15] on the two corpora to learn vertex embeddings. The aim is to capture the high-order proximity, which assumes that vertices frequently co-occurred in the same context of a sequence should be assigned to similar embeddings. Given a vertex sequence S and a vertex u_i, the context is defined as the ws vertices before u_i and after u_i in S; each vertex is associated with a context vector $\vec{\boldsymbol{\theta}}_i$ (or $\vec{\boldsymbol{\vartheta}}_j$) to denote its role as a context. As there are two types of vertices in a bipartite network, we preserve the high-order proximities separately. Specifically, for the corpus $\mathcal{D}^U$, the conditional probability

to maximize is:

$$\begin{aligned} maximize\ O_2 &= \prod_{u_i \in S \wedge S \in \mathcal{D}^U} \prod_{u_c \in C_S(u_i)} P(u_c|u_i). \\ maximize\ O_3 &= \prod_{v_j \in S \wedge S \in \mathcal{D}^V} \prod_{v_c \in C_S(v_j)} P(v_c|v_j). \end{aligned} \quad (5)$$

where $C_S(u_i)$ (or $C_S(v_j)$) denotes the context vertices of vertex u_i (or v_j) in a sequence S.

Following existing neural embedding methods [8], [14], [18], we parameterize the conditional probability $P(u_c|u_i)$ and $P(v_c|v_j)$ using the inner product kernel with softmax for output:

$$P(u_c|u_i) = \frac{\exp(\vec{\mathbf{u}_i}^T \vec{\boldsymbol{\theta}_c})}{\sum_{k=1}^{|U|} \exp(\vec{\mathbf{u}_i}^T \vec{\boldsymbol{\theta}_k})}, \quad P(v_c|v_j) = \frac{\exp(\vec{\mathbf{v}_j}^T \vec{\boldsymbol{\vartheta}_c})}{\sum_{k=1}^{|V|} \exp(\vec{\mathbf{v}_j}^T \vec{\boldsymbol{\vartheta}_k})}. \quad (6)$$

where $P(u_c|u_i)$ denotes how likely u_c is observed in the contexts of u_i; similar meaning applies to $P(v_c|v_j)$. With this definition, achieving the goal defined in Equations (5) will force the vertices with the similar contexts to be close in the embedding space. Nevertheless, optimizing the objectives is non-trivial, since each evaluation of the softmax function needs to traverse all vertices of a side, which is very time-costing. To reduce the learning complexity, we employ the idea of negative sampling [15].

3.2.3 *Negative Sampling*

The idea of negative sampling is to approximate the costly denominator term of softmax with some sampled negative instances [25]. Then the learning can be performed by optimizing a point-wise classification loss. For a center vertex u_i, high-quality negatives should be the vertices that are dissimilar from u_i. Towards this goal, some heuristics have been applied, such as frequency-based negative sampling method proposed to learn the representations for words [15]. Specifically, words with high frequency have large probability of being chosen as negative instances, which is suitable for language model since high frequency words are useless words such as *he, she, it, is, the*, etc. Nevertheless, high frequency vertices in bipartite networks are often the most important entities, such as popular items or active users, and the tracing phenomenon widely exiting in user buying and watching behaviors indicates us that it might be suboptimal to simply treat the high frequency vertices as negative instances. Here we propose a more grounded sampling method that caters the network data.

First we employ locality sensitive hashing (LSH) [26] to block vertices after shingling each vertex by its ws-hop neighbors with respect to the topological structure in the input bipartite network. Given a center vertex, we then randomly choose the negative samples from the buckets that are different from the bucket contained the center vertex. Through this way, we can obtain high-quality and diverse negative samples, since LSH can guarantee that dissimilar vertices are located in different buckets in a probabilistic way [26].

Let $N_S^{ns}(u_i)$ denote the ns negative samples for a center vertex u_i in sequence $S \in \mathcal{D}^U$, we can then approximate the conditional probability $p(u_c|u_i)$ defined in Equation (6) as:

$$p(u_c, N_S^{ns}(u_i)|u_i) = \prod_{z \in \{u_c\} \cup N_S^{ns}(u_i)} P(z|u_i), \quad (7)$$

where the probability $P(z|u_j)$ is defined as:

$$P(z|u_i) = \begin{cases} \sigma(\vec{\mathbf{u}_i}^T \vec{\boldsymbol{\theta}_z}), & \text{if } z \text{ is a context of } u_i \\ 1 - \sigma(\vec{\mathbf{u}_i}^T \vec{\boldsymbol{\theta}_z}), & z \in N_S^{ns}(u_i) \end{cases},$$

where σ denotes the sigmoid function $1/(1+e^{-x})$. By replacing $p(u_c|u_i)$ in Equation (5) with the definition of $p(u_c, N_S^{ns}(u_i)|u_i)$, we can get the approximated objective function to optimize. The semantics is that the proximity between the center vertex and their contextual vertices should be maximized, whereas the proximity between the center vertex and the negative samples should be minimized.

Following the similar formulations, we can get the counterparts for the conditional probability $p(v_c|v_j)$, the details of which are omitted here due to space limitation.

3.3 Joint Optimization

To embed a bipartite network by preserving both explicit and implicit relations simultaneously, we combine their objective functions to form a joint optimization framework.

$$maximize\ L = \alpha \log O_2 + \beta \log O_3 - \gamma O_1. \quad (8)$$

where parameters α, β and γ are hyper-parameters to be specified to combine different components in the joint optimization framework.

To optimize the joint model, we utilize the Stochastic Gradient Ascent algorithm (SGA). Note that the three components of Equation (8) have different definitions of a training instance. To handle this issue, we tweak the SGA algorithm by performing a gradient step as follows:

Step I: For a stochastic explicit relation, i.e., an edge $e_{ij} \in E$, we first update the embedding vectors $\vec{\mathbf{u}_i}$ and $\vec{\mathbf{v}_j}$ by utilizing SGA to maximize the last component $L_1 = -\gamma O_1$. We give the SGA update rule for $\vec{\mathbf{u}_i}$ and $\vec{\mathbf{v}_j}$ as follows:

$$\begin{aligned} \vec{\mathbf{u}_i} &= \vec{\mathbf{u}_i} + \lambda\{\gamma w_{ij}[1 - \sigma(\vec{\mathbf{u}_i}^T \vec{\mathbf{v}_j})] \cdot \vec{\mathbf{v}_j}\}, \\ \vec{\mathbf{v}_j} &= \vec{\mathbf{v}_j} + \lambda\{\gamma w_{ij}[1 - \sigma(\vec{\mathbf{u}_i}^T \vec{\mathbf{v}_j})] \cdot \vec{\mathbf{u}_i}\}. \end{aligned} \quad (9)$$

Step II: We then treat vertices u_i and v_j as the center vertex; by employing SGA to maximize objective functions $L_2 = \alpha \log O_2$ and $L_3 = \beta \log O_3$, we can preserve the implicit relations. Specifically, given the center vertex u_i (or v_j) and its context vertex u_c (or v_c), we update their embedding vectors $\vec{\mathbf{u}_i}$ (or $\vec{\mathbf{v}_j}$) as follows:

$$\begin{aligned} \vec{\mathbf{u}_i} &= \vec{\mathbf{u}_i} + \lambda\{\sum_{z \in \{u_c\} \cup N_S^{ns}(u_i)} \alpha[I(z, u_i) - \sigma(\vec{\mathbf{u}_i}^T \vec{\boldsymbol{\theta}_z})] \cdot \vec{\boldsymbol{\theta}_z}\}, \\ \vec{\mathbf{v}_j} &= \vec{\mathbf{v}_j} + \lambda\{\sum_{z \in \{v_c\} \cup N_S^{ns}(v_j)} \beta[I(z, v_j) - \sigma(\vec{\mathbf{v}_j}^T \vec{\boldsymbol{\vartheta}_z})] \cdot \vec{\boldsymbol{\vartheta}_z}\}. \end{aligned} \quad (10)$$

where $I(z, u_i)$ is an indicator function that determines whether vertex z is in the context of u_i or not; similar meaning applies to $I(z, v_j)$. Furthermore, the context vectors of both positive and negative instances are updated as:

$$\begin{aligned} \vec{\boldsymbol{\theta}_z} &= \vec{\boldsymbol{\theta}_z} + \lambda\{\alpha[I(z, u_i) - \sigma(\vec{\mathbf{u}_i}^T \vec{\boldsymbol{\theta}_z})] \cdot \vec{\mathbf{u}_i}\}, \\ \vec{\boldsymbol{\vartheta}_z} &= \vec{\boldsymbol{\vartheta}_z} + \lambda\{\beta[I(z, v_j) - \sigma(\vec{\mathbf{v}_j}^T \vec{\boldsymbol{\vartheta}_z})] \cdot \vec{\mathbf{v}_j}\}. \end{aligned} \quad (11)$$

We use the embedding vectors as the representations of vertices. Concatenating the embedding and contextual vectors for each vertex may improve the representations, which we leave as future work.

3.4 Discussion

Computational Complexity Analysis. The corpus generation and joint model optimization are two key processes of BiNE. Here we discuss the computational complexity of the two processes respectively.

For the large-scale network, the complexity of generating corpus will increase since $\mathbf{W}^U$ and $\mathbf{W}^V$ become large and dense. To avoid processing the dense matrix, we directly perform two-step walk in the original bipartite network to generate corpora. Let vc denotes the visitation count of vertex v in the generated corpus. The context size is therefore $vc \cdot 2ws$, which is a big value for vertices having high degrees. Yet we only randomly select a small batch, e.g., bs ($bs \ll vc$), of the contextual vertices for each center vertex. Thus, the complexity of algorithm is $O(2|E| \cdot bs \cdot 2ws \cdot (ns+1))$. To some extent, all the contextual vertices of a center vertex can be trained in each iteration by setting a proper bs, because the center vertex will be visited more than once when traversing all edges. Consequently, the performance is also guaranteed while the executive efficiency of BiNE is greatly improved.

4 BiNE as Factorizing Multiple Matrices

Prior efforts have revealed that several network embedding methods like DeepWalk, LINE and node2vec can be understood as performing factorization on some purposefully designed matrices [27]. The fundamental reason is that these methods use inner product to measure the affinity of two vertices, which also forms the basis of matrix factorization [28]. In this section, we prove that the BiNE embeddings are optimized towards the objective of factorizing multiple matrices, establishing the connections between BiNE and matrix factorization.

4.1 Derivation of the Implicit Matrices

In the following analysis, let $\#(u_i, u_j)$ (or $\#(v_i, v_j)$) be the number of center-context vertex pairs (u_i, u_j) (or (v_i, v_j)) in $\mathcal{D}^U$ (or $\mathcal{D}^V$). Moreover, $\#(u_i) = \sum_{j=1}^{|U|} \#(u_i, u_j)$ (or $\#(v_i) = \sum_{j=1}^{|V|} \#(v_i, v_j)$) denotes the number of times that vertex u_i (or v_i) appeared in $\mathcal{D}^U$ (or $\mathcal{D}^V$).

4.1.1 Global Objective Functions

The optimization of $\log O_2$ in Equation (8) is trained in an online fashion via stochastic gradient updates over the observed pairs (u_i, u_j) in the corpus $\mathcal{D}^U$. Its global objective can be obtained by summing over the observed (u_i, u_j) pairs in the corpus [29]:

$$maximize \ \log O_2 = \sum_{i=1}^{|U|} \sum_{j=1}^{|U|} \#(u_i, u_j) \cdot \ell_{uu}(i, j). \tag{12}$$

where

$$\ell_{uu}(i, j) = \log \sigma(\vec{\mathbf{u}}_i^T \vec{\boldsymbol{\theta}}_j) + \sum_{j' \in N_S^{ns}(u_i)} \log \sigma(-\vec{\mathbf{u}}_i^T \vec{\boldsymbol{\theta}}_{j'}), \tag{13}$$

which is the local objective function for a single center-context vertex pair (u_i, u_j).

Let $\vec{\mathbf{u}}_i^T \vec{\boldsymbol{\theta}}_j \doteq x_{ij}^U$, then the local objective function $\ell_{uu}(i, j)$ can be treated as a function of x_{ij}^U, in which the value of each term is mutually independent. Therefore, $\ell_{uu}(i, j)$ can be simplified as:

$$\ell_{uu}(i, j) = \log \sigma(x_{ij}^U) + \sum_{j' \in N_S^{ns}(u_i)} \log \sigma(-x_{ij'}^U). \tag{14}$$

In BiNE, for each center vertex u_i, its negative sample $u_{j'}$ is uniformly sampled from $N_S^{ns}(u_i)$. Thus, based on the importance sampling, the second term of Equation (14) can be approximated by the conditional expectation given $u_{j'} \in N_S^{ns}(u_i)$.

$$\begin{aligned} \sum_{j' \in N_S^{ns}(u_i)} \log \sigma(-x_{ij'}^U) &= \sum_{j' \in N_S^{ns}(u_i)} \frac{\log \sigma(-x_{ij'}^U)}{p(j')} \cdot p(j') \\ &\approx E_{j' \sim p}\big[\frac{\log \sigma(-x_{ij'}^U)}{p(j')} | N_S^{ns}(u_i)\big] \\ &= ns \cdot E_p\big[\log \sigma(-x_{ij'}^U) | N_S^{ns}(u_i)\big]. \end{aligned} \tag{15}$$

where $p(j') = \frac{1}{ns}$ is a conditional probability when $u_{j'}$ samples from $N_S^{ns}(u_i)$.

Unfortunately, the conditional expectation is a random variable related to the center vertex u_i, rather than a constant. As a result, the log-likelihood $\log O_2$ in Equation (12) cannot be maximized directed. Therefore, we employ the EM-algorithm to maximize the log-likelihood $\log O_2$.

E-Step: The expectation of $\ell_{uu}(i, j)$ is:

$$\begin{aligned} E[\ell_{uu}(i, j)] &= \log \sigma(x_{ij}^U) + ns \cdot E\big[E_p\big[\log \sigma(-x_{ij'}^U) | N_S^{ns}(u_i)\big]\big] \\ &= \log \sigma(x_{ij}^U) + ns \cdot E\big[\log \sigma(-x_{ij'}^U)\big]. \end{aligned} \tag{16}$$

As mentioned in Section 3.2.3, the LSH-based negative sample $u_{j'}$ is sampled from the bucket that is different from the bucket contained the center vertex u_i. Let $\mathcal{J}_{ij}$ be the Jaccard similarity between vertices u_i and u_j, then the probability, that u_i and u_j are mapped into the different buckets, can be computed as $q(u_i, u_j) = (1-(\mathcal{J}_{ij})^k)^b$, where b and k denote the number of # bands and the number of # rows in each band, which are two hyper-parameters in the LSH. It is noteworthy that the vertices are mapped into the same bucket of LSH with larger probability if they have a higher proximity. That is, given the center vertex u_i, a context vertex is mapped to the different buckets with a small probability. Since both $q(u_i, u_j)$ and the number of context vertices are small, $E[\ell_{uu}(i, j)]$ can be approximated as:

$$\begin{aligned} E[\ell_{uu}(i, j)] &= \log \sigma(x_{ij}^U) + ns \cdot \sum_{j' \in N_S(u_i)} q(u_i, u_{j'}) \cdot \log \sigma(-x_{ij'}^U) \\ &\approx \log \sigma(x_{ij}^U) + ns \cdot \sum_{j=1}^{|U|} q(u_i, u_j) \cdot \log \sigma(-x_{ij}^U) \end{aligned} \tag{17}$$

Naturally, the expectation of the log-likelihood function $\log O_2$ can be approximated as:

$$\begin{aligned} &E[\log O_2] \\ &= \sum_{i=1}^{|U|} \sum_{j=1}^{|U|} \#(u_i, u_j) \cdot E[\ell_{uu}(i, j)] \\ &= \sum_{i=1}^{|U|} \sum_{j=1}^{|U|} \#(u_i, u_j) \cdot [\log \sigma(x_{ij}^U) + ns \cdot \sum_{l=1}^{|U|} q(u_i, u_l) \cdot \log \sigma(-x_{il}^U)] \\ &= \sum_{i=1}^{|U|} \sum_{j=1}^{|U|} [\#(u_i, u_j) \cdot \log \sigma(x_{ij}^U) + ns \cdot \#(u_i) \cdot q(u_i, u_j) \cdot \log \sigma(-x_{ij}^U)]. \end{aligned} \tag{18}$$

$$
\begin{aligned}
&E[\log O_2] \\
&= \sum_{i=1}^{|U|}\sum_{j=1}^{|U|} \#(u_i,u_j)\cdot E[\ell_{uu}(i,j)] \\
&= \sum_{i=1}^{|U|}\sum_{j=1}^{|U|} \#(u_i,u_j)\cdot [\log\sigma(x_{ij}^U) + ns\cdot \sum_{l=1}^{|U|} q(u_i,u_l)\cdot\log\sigma(-x_{il}^U)] \\
&= \sum_{i=1}^{|U|}\sum_{j=1}^{|U|} [\#(u_i,u_j)\cdot\log\sigma(x_{ij}^U) + ns\cdot\#(u_i)\cdot q(u_i,u_j)\cdot\log\sigma(-x_{ij}^U)]
\end{aligned} \tag{19}
$$

The optimal solution $(x_{ij}^U)^*$ can be obtained if we maximize the expectation of log-likelihood $\log O_2$.

$$
\frac{\partial E[\ell_{uu}]}{\partial x_{ij}^U} = \#(u_i,u_j)\cdot\sigma(-x_{ij}^U) - ns\cdot\#(u_i)\cdot q(u_i,u_j)\cdot\sigma(x_{ij}^U). \tag{20}
$$

Moreover, the solution of $\frac{\partial E[\ell_{uu}]}{\partial x_{ij}^U} = 0$ can be calculated in a closed form as follows:

$$
(x_{ij}^U)^* = \log\frac{\#(u_i,u_j)}{\#(u_i)\cdot ns\cdot q(u_i,u_j)}. \tag{21}
$$

In Equation (21), $(x_{ij}^U)^*$ can be rewritten as

$$
\log\frac{\#(u_i,u_j)}{\#(u_i)\cdot q(u_i,u_j)} - \log ns, \tag{22}
$$

where $q(u_i,u_j)$ is the probability that u_i and u_j are mapped into the different buckets in the LSH, and

$$
\frac{\#(u_i,u_j)}{\#(u_i)} = \frac{\#(u_i,u_j)}{|\mathcal{D}|}\cdot\frac{|\mathcal{D}|}{\#(u_i)} = \frac{p(u_i,u_j)}{p(u_i)} = p(u_j|u_i) \tag{23}
$$

is the conditional probability, which can represent the proximity of center-context pair (u_i,u_j) in the corpus $\mathcal{D}$. $(x_{ij}^U)^*$ tends to be a larger value if u_i and u_j are vertex pair with a higher proximity. This is due to the fact that $p(u_j|u_i)$ is larger and $q(u_i,u_j)$ is smaller in this case.

Similarly, we can obtain the optimal solution of (x_{ij}^V) via maximizing $\log O_3$:

$$
(x_{ij}^V)^* = \log\frac{\#(v_i,v_j)}{\#(v_i)\cdot ns\cdot q(v_i,v_j)}. \tag{24}
$$

4.1.2 Approximating the Implicit Matrices

Based on the above analysis, we obtain the maximum likelihood estimation (MLE) of $\log O_2$ by setting $\vec{\mathbf{u}_i}^T\vec{\boldsymbol{\theta}_j} = (x_{ij}^U)^*$ for every center-context pair (u_i,u_j). Let $\mathbf{M}_{ij}^U = (x_{ij}^U)^*$ be a weight which evaluates how similar between u_i and u_j in the embedding space. To learn the representation of vertex u_i, we only need to factorize the matrix $\mathbf{M}^U$. Thus, the vertex learning problem is cast as a matrix factorization issue. For learning the representation of vertex v_i, it can be done in the same manner.

Note that,

$$
\begin{aligned}
\mathbf{M}_{ij}^U &= (x_{ij}^U)^* = \vec{\mathbf{u}_i}^T\vec{\boldsymbol{\theta}_j} = \log\tfrac{\#(u_i,u_j)}{\#(u_i)\cdot ns\cdot q(u_i,u_j)}, \\
\mathbf{M}_{ij}^V &= (x_{ij}^V)^* = \vec{\mathbf{v}_i}^T\vec{\boldsymbol{\vartheta}_j} = \log\tfrac{\#(v_i,v_j)}{\#(v_i)\cdot ns\cdot q(v_i,v_j)}.
\end{aligned} \tag{25}
$$

As indicated in the above equations, we need to generate the vertex sequences for deriving the matrices $\mathbf{M}^U$ and $\mathbf{M}^V$. As illustrated in subsection 5.3, the parameters for generating vertex sequences are dependent on the size of dataset. Thus, they are difficult to tune in practice. To avoid generating the vertex sequences, we approximate $\frac{\#(u_i,u_j)}{\#(u_i)}$ and $\frac{\#(v_i,v_j)}{\#(v_i)}$ according to the following theories.

Lemma 1. *Let $\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{W}$ be the transition matrix of a random walk, where $\boldsymbol{W}$ is the weighted matrix of a homogeneous network G; $\boldsymbol{D} = diag(d_1,..d_i,...)$, where d_i represents the weighted degree of vertex i; $vol(G) = \sum_i\sum_j w_{ij}$; $L_{\mathcal{D}}$ is the length of corpus $\mathcal{D}$ generated by the random walk. When $L_{\mathcal{D}}\to\infty$, $\frac{\#(w,c)}{|\mathcal{D}|}$ converges in probability ($\xrightarrow{p}$) as follows:*

$$
\frac{\#(w,c)}{|\mathcal{D}|} \xrightarrow{p} \frac{1}{2\cdot ws}\sum_{r=1}^{ws}\left(\frac{d_w}{vol(G)}(\boldsymbol{P}^r)_{w,c} + \frac{d_c}{vol(G)}(\boldsymbol{P}^r)_{c,w}\right). \tag{26}
$$

$$
\frac{\#(w)}{|\mathcal{D}|} \xrightarrow{p} \frac{d_w}{vol(G)}, \quad \frac{\#(c)}{|\mathcal{D}|} \xrightarrow{p} \frac{d_c}{vol(G)}. \tag{27}
$$

where ws is the window size, w is the center vertex and c is a context vertex of w. $\#(w,c)$, $\#(w)$ and $\#(c)$ denote the number of times that the center-context pair (w,c), center vertex w and context c appear in the corpus, respectively.

Lemma 2. *Suppose that $f(x)$ is a continuous function, and*

$$
\eta_n \xrightarrow{p} \eta, \quad \xi_n \xrightarrow{p} \xi. \tag{28}
$$

Then, we have

$$
f(\eta_n) \xrightarrow{p} f(\eta), \quad \xi_n\eta_n \xrightarrow{p} \xi\eta. \tag{29}
$$

Theorem 1. *Let $\boldsymbol{P}$ be the transition matrix of a random walk, and $\mathcal{D}$ be the generated corpus. When $L_{\mathcal{D}}\to\infty$, we have*

$$
\frac{\#(w,c)}{\#(w)} \xrightarrow{p} \frac{1}{ws}\sum_{r=1}^{ws}(\boldsymbol{P}^r)_{w,c}. \tag{30}
$$

Proof. Since $f(x) = \frac{1}{x}$ is a continuous function, thus

$$
\frac{|\mathcal{D}|}{\#(w)} \xrightarrow{p} \frac{vol(G)}{d_w}. \quad \text{(Lemma 2)}
$$

According to the Lemma 1 and 2, we have

$$
\begin{aligned}
\frac{\#(w,c)}{\#(w)} &\xrightarrow{p} \frac{1}{2\cdot ws}\sum_{r=1}^{ws}\frac{vol(G)}{d_w}\left(\frac{d_w}{vol(G)}(\mathbf{P}^r)_{w,c} + \frac{d_c}{vol(G)}(\mathbf{P}^r)_{c,w}\right) \\
&= \frac{1}{2\cdot ws}\left(\sum_{r=1}^{ws}(\mathbf{P}^r)_{w,c} + \frac{d_c}{d_w}\sum_{r=1}^{ws}(\mathbf{P}^r)_{c,w}\right).
\end{aligned} \tag{31}
$$

Moreover, the above equation can be rewritten in the matrix form as follows:

$$
\begin{aligned}
&\frac{1}{2\cdot ws}\left(\sum_{r=1}^{ws}(\mathbf{P}^r) + \sum_{r=1}^{ws}\mathbf{D}^{-1}(\mathbf{P}^r)^T\mathbf{D}\right) \\
&= \frac{1}{2\cdot ws}\sum_{r=1}^{ws}\underbrace{\mathbf{D}^{-1}\mathbf{W}\mathbf{D}^{-1}\mathbf{W}...\mathbf{D}^{-1}\mathbf{W}}_{r} \\
&+ \frac{1}{2\cdot ws}\sum_{r=1}^{ws}\mathbf{D}^{-1}\underbrace{\mathbf{W}\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}...\mathbf{W}\mathbf{D}^{-1}}_{r}\mathbf{D} \\
&= \frac{1}{2\cdot ws}\left(\sum_{r=1}^{ws}(\mathbf{P}^r) + \sum_{r=1}^{ws}(\mathbf{P}^r)\right) = \frac{1}{ws}\sum_{r=1}^{ws}(\mathbf{P}^r).
\end{aligned} \tag{32}
$$

Thus, we can obtain $\frac{\#(w,c)}{\#(w)} \xrightarrow{p} \frac{1}{ws}\sum_{r=1}^{ws}(\mathbf{P}^r)_{w,c}$. □

Using the Theorem 1 on BiNE, we have

$$\frac{\#(u_i,u_j)}{\#(u_i)} \xrightarrow{p} \frac{1}{ws}\sum_{r=1}^{ws}(\mathbf{P}^U)^r_{ij},$$
$$\frac{\#(v_i,v_j)}{\#(v_i)} \xrightarrow{p} \frac{1}{ws}\sum_{r=1}^{ws}(\mathbf{P}^V)^r_{ij}. \quad (33)$$

where $\mathbf{P}^U$ and $\mathbf{P}^V$ are the transition matrix of the two homogeneous networks in BiNE. Further, the optimal solution $(x^U)^*$ and $(x^V)^*$ are replaced as:

$$(x^U_{ij})^* = \log\frac{1}{ws\cdot ns\cdot q(u_i,u_j)}\sum_{r=1}^{ws}(\mathbf{P}^U)^r_{ij},$$
$$(x^V_{ij})^* = \log\frac{1}{ws\cdot ns\cdot q(v_i,v_j)}\sum_{r=1}^{ws}(\mathbf{P}^V)^r_{ij}. \quad (34)$$

Finally, the matrix $\mathbf{M}^U$ and $\mathbf{M}^V$ are approximated as:

$$\mathbf{M}^U_{ij} = (x^U_{ij})^* = \vec{\mathbf{u}_i}^T\vec{\boldsymbol{\theta}_j} = \log\frac{1}{ws\cdot ns\cdot q(u_i,u_j)}\sum_{r=1}^{ws}(\mathbf{P}^U)^r_{ij},$$
$$\mathbf{M}^V_{ij} = (x^V_{ij})^* = \vec{\mathbf{v}_i}^T\vec{\boldsymbol{\vartheta}_j} = \log\frac{1}{ws\cdot ns\cdot q(v_i,v_j)}\sum_{r=1}^{ws}(\mathbf{P}^V)^r_{ij}. \quad (35)$$

However, the matrices $\mathbf{M}^U$ and $\mathbf{M}^V$ are not only ill-defined since $\log 0 = -\infty$, but also they are dense, which leads to a computational challenge of factoring the matrices $\mathbf{M}^U$ and $\mathbf{M}^V$ by element-wise algorithms. Inspiring by the shifted PPMI approach [29], we define $\mathbf{M'}^R_{ij} = \max(\mathbf{M}^R_{ij}, 0)$, where $R \in \{U, V\}$. In this way, $\mathbf{M'}^U$ and $\mathbf{M'}^V$ become sparse matrices.

4.2 Co-factorizing Multiple Matrices

Overall, the proposed BiNE algorithm with LSH-based negative sampling can be transferred into another one in a closed form, namely BiNE-MF, which learns the vertex representation via employing the matrix factorization.

In detail, BiNE-MF can be regarded as jointly factorizing three matrices: the weighted matrix $\mathbf{W}$ and two complement matrices $\mathbf{M'}^U$, $\mathbf{M'}^V$, where $\mathbf{W}$ preserves the explicit relations in the bipartite network, and both $\mathbf{M'}^U$ and $\mathbf{M'}^V$ preserve the implicit relations in the bipartite network. As shown in Fig. 1, the matrix $\mathbf{M'}^U$ shares the vertex embedding matrix of $\mathbf{U}$ with $\mathbf{W}$ and the matrix $\mathbf{M'}^V$ shares the vertex embedding matrix of $\mathbf{V}$ with $\mathbf{W}$. We set $\mathbf{H}$, $\mathbf{A}$, $\mathbf{B}$ as

$$\mathbf{H} = \begin{pmatrix}\mathbf{W} & \alpha'\mathbf{M'}^U\\ \beta'\mathbf{M'}^V & \mathbf{O}\end{pmatrix}, \mathbf{A} = \begin{pmatrix}\mathbf{U}\\ \beta'\mathbf{V'}\end{pmatrix}, \mathbf{B} = \begin{pmatrix}\mathbf{V}^T & \alpha'\mathbf{U'}^T\end{pmatrix}$$

where $\mathbf{O}$ can be any matrix (i.e., loss will be zero when matrix $\mathbf{O}$ is reconstructed), α' and β' are scaling parameters to balance the importance of explicit and implicit relations. Interestingly, matrix $\alpha'\mathbf{U'} = [\vec{\boldsymbol{\theta}_i}]$ and $\beta'\mathbf{V'} = [\vec{\boldsymbol{\vartheta}_j}]$ can be viewed as the context embedding matrices of U and V, respectively.

Once we obtain the matrix $\mathbf{H}$, we can use symmetric SVD [29] or SMF (stochastic matrix factorization) [30] to map the vertices into a low-dimensional space and obtain the embedding matrices $\mathbf{U}$ and $\mathbf{V}$.

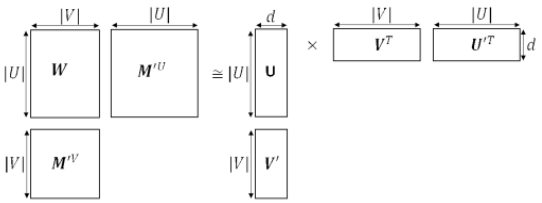


Fig. 1: An example of implicitly factorizing multiple matrices

4.3 Discussions

4.3.1 Effect of Negative Sampling

As mentioned in Equation (23), LSH-based negative sampling method used in BiNE is implicitly factorizing

$$\log\frac{\#(w,c)}{\#(w)\cdot q(w,c)} - \log ns. \quad (36)$$

where $q(w,c)$ is the probability that w and c locate in the different buckets of LSH. In contrast, the frequency-based negative sampling method in SGNS [29] is implicitly factorizing

$$\log\frac{\#(w,c)\cdot|\mathcal{D}|}{\#(w)\cdot\#(c)} - \log ns. \quad (37)$$

Here, $\frac{\#(w,c)}{\#(w)} = p(c|w)$ represents the proximity of center-context pair (w,c) in the corpus $\mathcal{D}$. Moreover, both $q(w,c)$ and $\frac{\#(c)}{|\mathcal{D}|}$ can be treated as penalty factors which contribute to measure the proximity of center vertex with its context vertices.

Assume that w and c have higher proximity, i.e., $P(c|w)$ is larger. Then, $q(w,c)$ used in BiNE will be smaller due to the lower probability of mapping w and c into different buckets. In this case, $\log\frac{\#(w,c)}{\#(w)\cdot q(w,c)} - \log ns$ will be larger. However, $\frac{\#(c)}{|\mathcal{D}|}$ used in SGNS is undetermined since its value can be large or small. Furthermore, the large value of $\log\frac{\#(w,c)\cdot|\mathcal{D}|}{\#(w)\cdot\#(c)} - \log ns$ biases c towards infrequent vertices, which may be inconsistent with our intuition.

4.3.2 Computational Complexity Analysis

Computing the matrices $\mathbf{M'}^U$ (or $\mathbf{M'}^V$) directly is a challenging task when the homogeneous network is large and dense or the window size is large. We resort to the approximation algorithm proposed in the work [27] to reduce the computation complexity. Unlike the online training method BiNE, the MF-based methods, such as SMF and BiNE-MF, work over aggregated center-context pair statistics using $(u_i, v_j w_{ij})$, $(u_i, u_j, \mathbf{M'}^U_{ij})$ and $(v_i, v_j, \mathbf{M'}^V_{ij})$ triples as input, which makes the optimization more directly and salable to large bipartite networks.

5 Experiments

In this section, we perform experiments on real-world datasets with the aim of answering the following research questions:

RQ1 How does BiNE perform compared with state-of-the-art network embedding methods?

RQ2 Can the MF-based methods, such as symmetric SVD, SMF, and BiNE-MF, achieve similar performance as that of BiNE?

RQ3 Can our proposed random walk generator contribute to preserving the long-tail distribution of vertex in bipartite networks and the implicit relations mined by it be helpful to learn better vertex representations?

RQ4 Does LSH-based negative sampling strategies superior to the frequency-based strategies in modeling bipartite networks?

The following sections will illustrate the experimental settings before answering the above research questions. In addition, a case study that visualizes a small bipartite network is performed to demonstrate the rationality of BiNE.

5.1 Experimental Settings

5.1.1 *Datasets*

We purposefully chosen unweighted networks for link prediction, which is usually approached as a classification task that predicts whether a link exists between two vertices; while we use weighted networks for recommendation task, which is a personalized ranking task that aims to provide items of interest for a user.

1) Unweighted bipartite network. We construct two unweighted bipartite networks from Wikipedia and Tencent, respectively. Specifically, the Wikipedia dataset contains the edit relationship between authors and pages, which is public accessible[1]; The Tencent dataset records the watching behaviors of users on movies in QQlive[2] in one month's time.
2) Weighted bipartite network. We construct other three weighted bipartite network from DBLP, Movielens and VisualizeUs, respectively. Specifically, the DBLP[3] dataset contains the publish relationship between authors and venues, where the edge weight indicates the number of papers published on a venue by an author; The Movielens[4] dataset records the rating behavior of users on movies, where the edge weight denotes the rating score of a user on a movie; The VisualizeUs[5] dataset records the tagging behavior of users on pictures, where the edge describes the number of times of tagging of a user on a picture.

The statistics of our experimented datasets are summarized in Table 1.

TABLE 1: Statistics of bipartite networks and metrics adopted in experiments for different tasks.

Task	Link Prediction		Recommendation		
Type	undirected, unweighted		undirected, weighted		
Metric	AUC-ROC,AUC-PR		F1, NDCG, MAP, MRR		
Name	Tencent	Wikipedia	VisualizeUs	DBLP	MovieLens
$\|U\|$	14,259	15,000	6,000	6,001	69,878
$\|V\|$	1,149	3,214	3,355	1,308	10,677
$\|E\|$	196,290	172,426	35,639	29,256	10,000,054
Density	1.2%	0.4%	0.2%	0.4%	1.3%

5.1.2 *Evaluation Protocols*

1) For link prediction task, we first apply the same protocol as the Node2vec paper [14] to process the Wikipedia dataset. Specifically, we first treat the observed links as the positive instances, and sample an equal number of unconnected vertex pairs as the negative instances. For Tencent dataset, we treat the user-movie pairs as positive instances if the user has watched the movie for more than 5 minutes, otherwise, negative instances. For both datasets, we randomly sample 60% instances as the training set, and evaluate the link prediction performance on the remaining 40% dataset with two metrics: the ROC curve (AUC-ROC) and Precision-Recall curve (AUC-PR).
2) For recommendation task, we randomly sample 60% edges as the training data, and the remaining 40% edges are treated as testing dataset for all datasets. We rank all items in the testing set for each user and truncate the ranking list at 10 to evaluate the performance of top-10 recommendation with four IR metrics: F1, Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR).

For each metric, we compute the average score for all users, and perform paired sample T-test on it. To avoid overfitting, we generate 10 folds of train-test split, tuning hyper-parameters on the first fold only for each method. We use the optimal hyper-parameter setting and report the average performance of all folds (i.e., the score of each metric and the p-value of t-test).

5.1.3 *Baselines*

We compare BiNE with three types of baselines:

1) **Network Embedding Methods**. We chose four representative of state-of-the-art network embedding methods as our baselines, including homogeneous and heterogeneous network embedding methods. For each method, we use the released implementations for our experiments.
   - DeepWalk [8]: This method performs uniform random walks to get a corpus of vertex sequences. Then the word2vec is applied on the corpus to learn vertex embeddings for homogeneous networks.
   - Node2vec [14]: This approach extends DeepWalk by performing biased random walks to generate the corpus of vertex sequences. The hyper-parameters p and q are set to 0.5 which has empirically shown good results.
   - LINE [18]: In contrast to the above methods, this approach optimizes both the 1st-order and 2nd-order proximities in a homogeneous network without generating corpus. We use the LINE (1st+2nd) method which has shown the best performance in their paper.
   - Metapath2vec++ [11]: As a state-of-the-art method for embedding heterogeneous networks, it generates corpus following the predefined meta-path scheme. And the meta-path scheme chosen in our experiments are "IUI" (item-user-item) and "IUI"+"UIU" (user-item-user), and we only report the best result between them.
2) We compare with a set of methods that are specifically designed for the link prediction task. We apply several indices proposed in [31], including Absent Links (AL), Katz Index (Katz), and Preferential Attachmenthave (PA).
3) We also compare with several competitive methods[6] that are designed for the top-K item recommendation task.

1. http://konect.uni-koblenz.de/networks/wikipedia_link_en
2. https://v.qq.com/
3. http://dblp.uni-trier.de/xml/
4. http://grouplens.org/datasets/movielens/
5. http://konect.uni-koblenz.de/networks/pics_ti
6. We use the LibRec implementation: https://www.librec.net/

- BPR [32]:This method has been widely used in recommendation literature as a highly competitive baseline [28]. It optimizes the matrix factorization (MF) model with a pairwise ranking-aware objective.
- RankALS [33]: This method also optimizes the MF model for the ranking task, by towards a different pairwise regression-based loss.
- FISMauc [34]: Distinct to MF, factored item similarity model (FISM) is an item-based collaborative filtering method. We employ the AUC-based objective to optimize FISM for the top-K task.

TABLE 2: The search range and optimal setting (highlighted in red) of hyper-parameters for our BiNE method.

Parameter	Meaning	Test values
ns	number of negative samples	[1, 2, **4**, 6, 8, 10]
ws	size of window	[1, 3, **5**, 7, 9]
p	walk stopping probability	[0.05, 0.1, **0.15**, 0.2, 0.3, 0.4, 0.5]
β	trade-off parameter	[0.0001, 0.001, **0.01**, 0.1, 1]
γ	trade-off parameter	[0.01, 0.05, **0.1**, 0.5, **1**, 5]
α'	trade-off parameter	[0.0001, **0.001**, **0.01**, **0.1**, 1]
β'	trade-off parameter	[0.0001, **0.001**, **0.01**, **0.1**, 1]

5.1.4 *Parameter Settings*

We have fairly tuned the hyper-parameters for each method. For all network embedding methods, we set the embedding size as 128 for a fair comparison; other hyper-parameters follow the default setting of their released implementations. For the recommendation baselines, we tuned the learning rate and latent factor number since they impact most on the performance; other hyper-parameters follow the default setting of the LibRec toolkit.

For BiNE, we fix the loss trade-off parameter α as 0.01 and tune the other two. The $minT$ and $maxT$ are respectively set to 1 and 32, which empirically show good results in most cases. We test the learning rate λ of [0.01, 0.025, 0.1]. And the optimal setting of learning rate is 0.025 for the VisualizeUs/DBLP dataset and 0.01 for others. The search range and optimal setting (highlighted in red font) of other parameters are shown in Table 2. Note that besides γ is set differently — 0.1 for recommendation and 1 for link prediction — other parameters are set to the same value for both tasks.

For different MF-based methods, we set α' and β' as the same values for pair comparison. And the optimal setting of α' and β' are 0.001 and 0.001 for VisualizeUs/DBLP/Wikipedia dataset, 0.01 and 0.01 for Tencent dataset and 0.1 and 0.1 for Movielens. We test the learning rate of [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01] for SMF and BiNE-MF.

5.2 Performance Comparison (RQ1)

5.2.1 *Link Prediction*

In this task, we first concatenate the embedding vectors $\vec{\mathbf{u}}_i$, $\vec{\mathbf{v}}_j$ and label (*i.e.,* the label of the positive instance is 1, otherwise 0) as a record for each instance (u_i, v_j) in the dataset, then feed the record into the logistic regression classifier with L2 loss function. Table 3 illustrates the performance of baselines and our BiNE, where we have the following key observations:

TABLE 3: Link prediction performance on Tencent and Wikipedia.

Algorithm	Tencent		Wikipedia	
	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR
AL	50.44%	65.70%	90.28%	91.81%
Katz	50.90%	65.06%	90.84%	92.42%
PA	55.60%	68.99%	90.71%	93.37%
DeepWalk	57.62%	71.32%	89.71%	91.20%
LINE	59.68%	73.48%	91.62%	93.28%
Node2vec	59.28%	72.62%	89.93%	91.23%
Metapath2vec++	60.70%	73.69%	89.56%	91.72%
BiNE	**60.98%****	**73.77%****	**92.91%****	**94.45%****

** indicates that the improvements are statistically significant for $p < 0.01$ judged by paired t-test.

- The neural network-based methods which trained in a data-dependent supervised manner outperform the indices proposed in [31] significantly.
- Metapath2vec++ and BiNE are significantly better than other neural network-based methods. This improvement demonstrates the positive effect of considering the information of node types when embedding bipartite networks.
- BiNE outperforms Metapath2vec++ significantly and achieves the best performance on both datasets in both metrics. This improvement points out the effectiveness of modeling of explicit and implicit relations in different ways.

5.2.2 *Recommendation*

In this task, we adopt the inner product kernel $\vec{\mathbf{u}}_i^T \vec{\mathbf{v}}_j$ to estimate the preference of user u_i on item v_j, and evaluate performance on the top-10 results. Table 4 shows the performance of baselines and our BiNE, where we have the following key observations:

- BiNE outperforms all baselines on all datasets, and the improvements are significant compared with Metapath2vec++ though it also considers the node type information when embedding bipartite networks. We hold that it is due to it ignores the weights and treats the two types of relations (*i.e.,* the explicit and implicit relations) as equally.
- BiNE outperforms LINE significantly though it also consider the weight information when embedding networks. The suboptimal performance obtained by LINE because of two reasons. (1) LINE ignores further high-order proximities among vertices due to it only preserves both 1st-order and 2nd-order relations when learning the representations for vertices. (2) LINE learns two separated embeddings for 1st-order and 2nd-order relations and concatenates them via post-processing, rather than optimizing them in a unified framework. Whereas BiNE mines high-order implicit relations among homogeneous vertices by performing random walks and design a joint framework to model the explicit and implicit relations jointly, where different relations reinforce each other and lead to better vertex representations.

5.3 Performance of MF-based Methods (RQ2)

Here we adopt symmetric SVD, stochastic matrix factorization (SMF) and BiNE-MF to obtain vertex embeddings

TABLE 4: Performance comparison of Top-10 recommendation on VisualizeUs, DBLP, and MovieLens.

Algorithm	VisualizeUs				DBLP				Movielens			
	F1@10	NDCG@10	MAP@10	MRR@10	F1@10	NDCG@10	MAP@10	MRR@10	F1@10	NDCG@10	MAP@10	MRR@10
BPR	6.22%	9.52%	5.51%	13.71%	8.95%	18.38%	13.55%	22.25%	8.03%	7.58%	2.23%	40.81%
RankALS	2.72%	3.29%	1.50%	3.81%	7.62%	11.50%	7.52%	14.87%	8.48%	7.95%	2.66%	38.93%
FISMauc	10.25%	15.46%	8.86%	16.67%	9.81%	13.77%	7.38%	14.51%	6.77%	6.13%	1.63%	34.04%
DeepWalk	5.82%	8.83%	4.28%	12.12%	8.50%	24.14%	19.71%	31.53%	3.73%	3.21%	0.90%	15.40%
LINE	9.62%	13.76%	7.81%	14.99%	8.99%	14.41%	9.62%	17.13%	6.91%	6.50%	1.74%	38.12%
Node2vec	6.73%	9.71%	6.25%	13.95%	8.54%	23.89%	19.44%	31.11%	4.16%	3.68%	1.05%	18.33%
Metapath2vec++	5.92%	8.96%	5.35%	13.54%	8.65%	25.14%	19.06%	31.97%	4.65%	4.39%	1.91%	16.60%
BiNE	**13.63%****	**24.50%****	**16.46%****	**34.23%****	**11.37%****	**26.19%****	**20.47%****	**33.36%****	**9.14%****	**9.02%****	**3.01%****	**45.95%****

** indicates that the improvements are statistically significant for $p < 0.01$ judged by paired t-test.

TABLE 5: Performance comparison of matrix factorization using different manners.

	Symmetric SVD		SMF		BiNE-MF		BiNE	
	Link Prediction							
Dataset	**AUC-ROC**	**AUC-PR**	**AUC-ROC**	**AUC-PR**	**AUC-ROC**	**AUC-PR**	**AUC-ROC**	**AUC-PR**
Tencent	62.58%	75.08%	**62.81%**	**75.19%**	61.39%	74.12%	60.98%	73.77%
WikiPedia	87.55%	90.93%	91.27%	93.68%	92.17%	**94.51%**	**92.91%****	94.45%
	Recommendation							
Dataset	**MAP@10**	**MRR@10**	**MAP@10**	**MRR@10**	**MAP@10**	**MRR@10**	**MAP@10**	**MRR@10**
VisualizeUs	2.47%	4.70%	14.26%	32.04%	16.27%	34.02%	**16.46%****	**34.23%****
DBLP	2.44%	5.33%	10.19%	21.54%	19.99%	**33.40%**	**20.47%****	33.36%
Movielens	0.33%	3.49%	2.25%	41.91%	**3.26%****	45.17%	3.01%	**45.95%****

** indicates that the improvements are statistically significant for $p < 0.01$ judged by paired t-test.

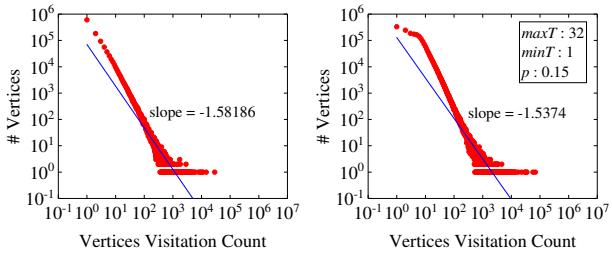
and compare their performance. The result is shown in Table 5. We have the following key observations:

- SMF and BiNE-MF that much like BiNE's training process show roughly equivalent performance with BiNE. They also obtain better performance than BiNE in the three big datasets: Tencent, WikiPedia, and Movielens. This reveals one advantage of MF-based methods that they approximate the global implicit relations while the limited scale of random walk negatively impacts BiNE's performance. And this observation is in line with the work [27]. It also indicates that $maxT$ should be specified to a large number for a large-scale network.
- Symmetric SVD yields the worser result than SMF and BiNE-MF in most case. The cause of this result is the matrix $\mathbf{H}$ is sparse and symmetric SVD is undefined when the matrix is incomplete. In addition, addressing the relatively few observed entries is highly prone to overfitting [30]. SMF and BiNE-MF are better than symmetric SVD at handling missing entries. And the regularization in SMF [29] and nonlinear variation in BiNE-MF are also two workable ways of improving performance.

5.4 Utility of Random Walk Generator (RQ3)

In this section, we first demonstrate the effectiveness of our random walk generator on preserving the characteristics of bipartite networks, especially the power-law distribution of vertices. Then, we illustrate the effect of considering the implicit relations when learning the representations of vertices for bipartite networks.

The frequency distribution of vertices in a real YouTube dataset is plotted in Fig. 2(a). We can see that the vertices exhibit a standard power-law distribution with a slope of -1.582. By contrast, we plot the frequency distribution of vertices in a corpus obtained from our random walk generator in Fig. 2(b). We can easily find that our random walk generator almost generates a standard power-law distribution with a slope -1.537 which is very close to that of the original network.



(a) YouTube (b) Random walk generator

Fig. 2: The vertex distribution of (a) the real-world YouTube dataset and (b) the corpus generated by our biased and self-adaptive random walk generator.

TABLE 6: BiNE with different random walk generators.

	Uniform Random Walk Generator		Biased and Self-adaptive Random Walk Generator	
	Link Prediction			
Dataset	**AUC-ROC**	**AUC-PR**	**AUC-ROC**	**AUC-PR**
Tencent	59.75%	73.06%	**60.98%****	**73.77%****
WikiPedia	88.77%	91.91%	**92.91%****	**94.45%****
	Recommendation			
Dataset	**MAP@10**	**MRR@10**	**MAP@10**	**MRR@10**
VisualizeUS	15.93%	33.66%	**16.46%****	**34.23%****
DBLP	11.79%	23.41%	**20.47%****	**33.36%****
MovieLens	2.91%	46.12%	**3.04%****	**46.20%****

** indicates that the improvements are statistically significant for $p < 0.01$ judged by paired t-test.

In addition, we compare the performance of BiNE under two settings — use or not use our proposed random walk generator. As shown in Table 6, the biggest absolute improvements of BiNE using our proposed random walk generator are 4.14% and 10.25% for link prediction and

TABLE 7: BiNE with and without implicit relations.

	Without Implicit Relations		With Implicit Relations	
Link Prediction				
Dataset	**AUC-ROC**	**AUC-PR**	**AUC-ROC**	**AUC-PR**
Tencent	59.78%	73.05%	**60.98%****	**73.77%****
WikiPedia	91.47%	93.73%	**92.91%****	**94.45%****
Recommendation				
Dataset	**MAP@10**	**MRR@10**	**MAP@10**	**MRR@10**
VisualizeUS	9.10%	19.76%	**16.46%****	**34.23%****
DBLP	20.20%	32.95%	**20.47%****	**33.36%****
MovieLens	2.86%	43.98%	**3.01%****	**45.95%****

** indicates that the improvements are statistically significant for $p < 0.01$ judged by paired t-test.

TABLE 8: BiNE with different negative sampling strategies.

	Frequency-based Negative Sampling		LSH-based Negative Sampling	
Link Prediction				
Dataset	**AUC-ROC**	**AUC-PR**	**AUC-ROC**	**AUC-PR**
Tencent	60.80%	73.64%	**60.98%**	**73.77%**
WikiPedia	92.21%	94.12%	**92.91%****	**94.45%****
Recommendation				
Dataset	**MAP@10**	**MRR@10**	**MAP@10**	**MRR@10**
VisualizeUS	15.07%	32.27%	**16.46%****	**34.23%****
DBLP	20.46%	32.93%	**20.47%**	**33.36%****
MovieLens	**3.01%**	45.86%	**3.01%**	**45.95%**

** indicates that the improvements are statistically significant for $p < 0.01$ judged by paired t-test.

recommendation, respectively. The above result indicates that the biased and self-adaptive random walk generator is helpful to capture the power-law distribution of vertices and contributes to improving the vertex representations for embedding bipartite networks. Please note that we change the value of $maxT$ to 128 for this empirical study on Movielens dataset because of the default value may be to small to fully preserve the implicit relations for such a large-scale bipartite network.

Lastly, we show the performance of BiNE and its variant which ignores the implicit relations. Due to the space limitation, we only show the performance on the recommendation task from two metrics: MAP@10 and MRR@10. From Table 7, we can find that the largest absolute improvements of BiNE with implicit relations are 1.44% and 18.58% for link prediction and recommendation, respectively. It demonstrates that our proposed way of mining high-order implicit relation as the complement of explicit relations is effect to modeling bipartite networks.

5.5 Negative Sampling Strategies (RQ4)

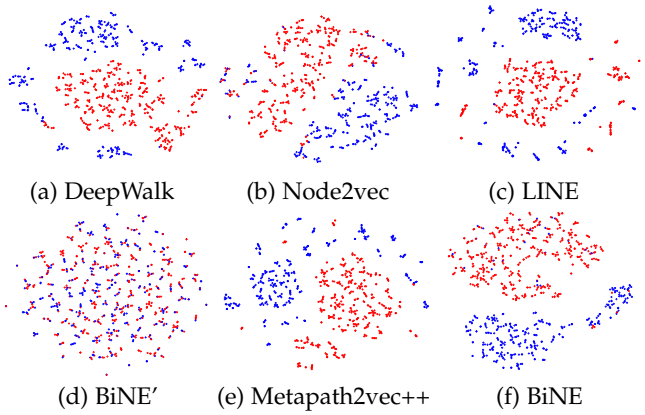
We have analyzed the difference between LSH-based and frequency-based negative sampling methods in section 4.3, that is LSH-based method resorts to dissimilar information deriving from observed links to obtain more accurate proximity of center-context vertex pairs, while frequency-based method utilizes frequency information to lower the proximity of center vertex with context vertices having high frequency. Here, we compare the performance of BiNE with different negative sampling strategies.

As shown in Table 8, there is a slight advantage in LSH-based by comparing it with frequency-based negative sampling method. Thus, we hold that LSH-based sampling method, which uses dissimilar information obtained from user behavior data, can generate more reasonable negative samples in modeling user behavior.

Frequency-based method also shows roughly equivalent performance in most cases. An intuitive explanation is that the number of negatives is large and the probability of sampling similar vertices as negative is small via frequency-based method strategies.

5.6 Case Study

In this section, we perform a visualization study for a small bipartite network to illustrate that the rationality of our BiNE method. We extra a small collaboration bipartite network from DBLP dataset, which contains 736 researchers and 6 international journals. A link will be established if the author has published at least 5 papers on the journal. The 6 journals are from two different research fields: SICOMP, IANDC and TIT from computer science theory, and AI, IJCV,and JMLR from artificial intelligence. As such, from the published venues of the researches, we can inference the research field of them.



(a) DeepWalk (b) Node2vec (c) LINE

(d) BiNE′ (e) Metapath2vec++ (f) BiNE

Fig. 3: Visualization of authors in DBLP. Color of a vertex indicates the research fields of the authors (red: "computer science theory", blue: "artificial intelligence"). BiNE′ is the version of BiNE – without implicit relations.

We utilize the t-SNE tool [35] to map the embedding vectors of authors into 2D space. In Fig. 3, we use different color to describe different research fields of the researchers, *i.e.*, red: "computer science theory", blue: "artificial intelligence", and show the visualization results given by different embedding approaches. From it, we can observe that DeepWalk, Node2vec, LINE, Metapath2vec++, and our BiNE are good since researchers belonging different research fields are well seperated. As far as we are concerned, BiNE gives a better result due to it generates an obvious gap between two research fields. However, the variant of BiNE ignoring the implicit relations – BiNE′ shows a worse layout than expected, which illustrate the effective of modeling high-order implicit relations for embedding bipartite networks.

6 Related Work

6.1 Network Representation Learning

Our work is related to the neural network-based NRL methods. We first review them from the perspective of network types.

The pioneer work DeepWalk [8] and Node2vec [14] extend the idea of Skip-gram [15] to model homogeneous network, which is convert to a corpus of vertex sequences by performing truncated random walks. However, they may not be effective to preserve both explicit and implicit relations of the network. There are some follow-up works exploiting both 1st-order and 2nd-order proximities between vertices to embed homogeneous networks. Specifically, LINE [18] learns two separated embeddings for 1st-order and 2nd-order relations; SDNE [36] and DVNE [7] incorporates both 1st-order and 2nd-order proximities to preserve the network structure; GraRep [37] and AROPE [38] further extends the method to capture higher-order proximities. Besides capturing high-order proximities, there are several proposals to incorporate side information into vertex embedding learning, such as vertex labels [10], [39], community information [40], textual content [41], user profiles [9], location information [42], among others. However, these methods might be suboptimal for learning vertex representations for a bipartite network by ignoring the vertex type information. In addition, the "corpus" generated by the truncated random walks may not capture the characteristics of the network structure, such as the power-law distribution of vertex degrees.

Metapath2vec++ [11] and HNE [12] and EOE [13] are representative vertex embedding methods for heterogeneous networks. Although they can be applied to bipartite network which can be seen as a special type of heterogeneous networks, they are not tailored for learning on bipartite networks. Specifically, HNE aims to integrate content and linkage structures into the embedding process, and Metapath2vec++ ignores the strength of the relations between vertices and treats the explicit and implicit relations as equally. As such, they are suboptimal for vertex representation learning for a bipartite network.

It is noteworthy that recent works have shown an increase of interest in generating vertex embedding by neighborhood aggregation encoders [43], [44]. However, these methods rely on vertex features or attributes.

6.2 Bipartite Network Modeling

As a ubiquitous data structure, bipartite networks have been mined for many applications, among which vertex ranking is an active research problem. For example, HITS [24] learns to rank vertices by capturing some semantic relations within a bipartite network. Co-HITS [22] incorporates content information of vertices and the constraints on relevance into vertex ranking of bipartite network. BiRank [16] ranks vertices by taking into account both the network structure and prior knowledge.

Distributed vertex representation is an alternative way to leverage signals from bipartite network. Unlike the ranking task, it learns a low dimensional representation of a vertex, which can be seen as the "features" of the vertex that preserves more information rather than simply a ranking score. Latent factor model (LFM), which has been widely investigated in the field of recommender systems and semantic analysis, is the most representative model. And a typical implementation of LFM is based on matrix factorization [32], [45], [46]. Recent advances utilize deep learning methods to learn vertex embeddings on the user-item network for recommendation [28]. It is worth pointing out that these methods are tailored for the recommendation task, rather than for learning informative vertex embeddings. Moreover, they model the explicit relations in bipartite network only, which can be improved by incorporating implicit relations as shown in [19], [20].

7 Conclusions

We have presented BiNE, a novel approach for embedding bipartite networks. It jointly models both the explicit relations and high-order implicit relations in learning the representation for vertices. Our theoretical result reveals that BiNE can be transferred into the algorithm BiNE-MF, which is a implicit multiple matrix factorization, in a closed form. As a result, it broadens the theoretical understanding of BiNE. Extensive experiments on several tasks of link prediction, recommendation, and visualization demonstrate the effectiveness and rationality of our BiNE method.

In this work, we have only considered the information revealed in observed edges, thus it may fail for vertices that have few or even no edges. Since missing data is a common situation in real-world applications, the observed edges may not contain sufficient signal on vertex relations. To address this issue, we plan to extend our BiNE method to model auxiliary side information, such as numerical features , textual descriptions , and among other attributes [9]. In addition, the bipartite networks in many practical applications are dynamically updated [46]. Thus, we plan to investigate how to efficiently refresh embeddings for dynamic bipartite networks.

8 Acknowledgments

This work has been supported by the National Key Research and Development Program of China under grant 2016YFB1000905, and the National Natural Science Foundation of China under Grant No. U1811264, 61877018, 61672234, and 61502236. NExT research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative , and the Shanghai Agriculture Applied Technology Development Program, China (Grant No.T20170303).

References

[1] Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. Bine: Bipartite network embedding. In *SIGIR*, pages 715–724, 2018.
[2] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. Item silk road: Recommending items from information domains to social users. In *SIGIR*, pages 185–194, 2017.
[3] Yixin Cao, Lifu Huang, Heng Ji, Xu Chen, and Juanzi Li. Bridge text and knowledge by learning multi-prototype entity mention embedding. In *ACL*, pages 1623–1633, 2017.
[4] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*, pages 1661–1670, 2015.

[5] Fuli Feng, Xiangnan He, Yiqun Liu, Liqiang Nie, and Tat-Seng Chua. Learning on partial-order hypergraphs. In *WWW*, pages 1523–1532, 2018.
[6] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *TKDE*, 2018.
[7] Dingyuan Zhu, Peng Cui, Daixin Wang, and Wenwu Zhu. Deep variational network embedding in wasserstein space. In *SIGKDD*, pages 2827–2836, 2018.
[8] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *KDD*, pages 701–710, 2014.
[9] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. Attributed social network embedding. *TKDE*, 2018.
[10] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. Attributed network embedding for learning in a dynamic environment. In *CIKM*, pages 387–396, 2017.
[11] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*, pages 135–144. ACM, 2017.
[12] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C. Aggarwal, and Thomas S. Huang. Heterogeneous network embedding via deep architectures. In *KDD*, pages 119–128, 2015.
[13] Linchuan Xu, Xiaokai Wei, Jiannong Cao, and Philip S. Yu. Embedding of embedding (EOE): joint embedding for coupled heterogeneous networks. In *WSDM*, pages 741–749, 2017.
[14] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864, 2016.
[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.
[16] Xiangnan He, Ming Gao, Min-Yen Kan, and Dingxian Wang. Birank: Towards ranking on bipartite graphs. *TKDE*, 29(1):57–71, 2017.
[17] Suchit Pongnumkul and Kazuyuki Motohashi. A bipartite fitness model for online music streaming services. *Physica A: Statistical Mechanics and its Applications*, 490:1125–1137, 2018.
[18] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: large-scale information network embedding. In *WWW*, pages 1067–1077, 2015.
[19] Lu Yu, Chuxu Zhang, Shichao Pei, Guolei Sun, and Xiangliang Zhang and. Walkranker: A unified pairwise ranking model with multiple relations for item recommendation. In *AAAI*, 2018.
[20] Meng Jiang, Peng Cui, Nicholas Jing Yuan, Xing Xie, and Shiqiang Yang. Little is much: Bridging cross-platform behaviors through overlapped crowds. In *AAAI*, pages 13–19, 2016.
[21] Taher Alzahrani, Kathy J. Horadam, and Serdar Boztas. Community detection in bipartite networks using random walks. In *CompleNet*, pages 157–165, 2014.
[22] Hongbo Deng, Michael R. Lyu, and Irwin King. A generalized co-hits algorithm and its application to bipartite graphs. In *KDD*, pages 239–248, 2009.
[23] Jure Leskovec, Deepayan Chakrabarti, Jon M. Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *JMLR*, 11:985–1042, 2010.
[24] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. In *ACM-SIAM*, pages 668–677, 1998.
[25] Hongzhi Yin, Lei Zou, Quoc Viet Hung Nguyen, Zi Huang, and Xiaofang Zhou. Joint event-partner recommendation in event-based social networks. In *ICDE*, 2018.
[26] Hongya Wang, Jiao Cao, LihChyun Shu, and Davood Rafiei. Locality sensitive hashing revisited: filling the gap between theory and algorithm analysis. In *CIKM*, pages 1969–1978, 2013.
[27] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*, pages 459–467, 2018.
[28] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.
[29] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185, 2014.
[30] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.
[31] Shuang Xia, Bing Tian Dai, Ee-Peng Lim, Yong Zhang, and Chunxiao Xing. Link prediction for bipartite social networks: The role of structural holes. In *ASONAM*, pages 153–157, 2012.
[32] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.
[33] Gábor Takács and Domonkos Tikk. Alternating least squares for personalized ranking. In *RecSys*, pages 83–90, 2012.
[34] Santosh Kabbur, Xia Ning, and George Karypis. FISM: factored item similarity models for top-n recommender systems. In *KDD*, pages 659–667, 2013.
[35] Van Der Maaten Laurens, Geoffrey Hinton, and Geoffrey Hinton, Van Der Maaten. Visualizing data using t-sne. *JMLR*, 9(2605):2579–2605, 2008.
[36] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *KDD*, pages 1225–1234, 2016.
[37] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *CIKM*, pages 891–900, 2015.
[38] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. Arbitrary-order proximity preserved network embedding. In *SIGKDD*, pages 2778–2786, 2018.
[39] Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *WSDM*, pages 550–558, 2017.
[40] Jifan Chen, Qi Zhang, and Xuanjing Huang. Incorporate group information to enhance network embedding. In *CIKM*, pages 1901–1904, 2016.
[41] Chuan-Ju Wang, Ting-Hsiang Wang, Hsiu-Wei Yang, Bo-Sin Chang, and Ming-Feng Tsai. Ice: Item concept embedding via textual information. In *SIGIR*, pages 85–94, 2017.
[42] Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang. Learning graph-based POI embedding for location-based recommendation. In *CIKM*, pages 15–24, 2016.
[43] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICRL*, 2016.
[44] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, pages 593–607, 2018.
[45] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. Discrete collaborative filtering. In *SIGIR*, pages 325–334, 2016.
[46] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*, pages 549–558, 2016.

Ming Gao is currently an associate professor on School of Data Science and Engineering at East China Normal University, China. He received his doctorate from the School of Computer Science, Fudan University. His research interests include knowledge engineering, user profiling, social network analysis and mining. His works appear in major international journals and conferences, including DMKD, TKDE, KAIS, ICDE, ICDM, SIGIR, etc.

Xiangnan He a senior research fellow with School of Computing, National University of Singapore (NUS). He received his Ph.D. in Computer Science from NUS. His research interests span recommender systems, information retrieval, and multi-media processing. He has over 50 publications appeared in several top conferences such as SIGIR, WWW, MM and IJCAI, and journals such as TKDE and TOIS. His work on recommender systems has received the Best Paper Award Honourable Mention of WWW 2018 and SIGIR 2016. Moreover, he has served as the PC member for several top conferences including SIGIR, WWW, MM, KDD etc., and the regular reviewer for journals including TKDE, TOIS, TMM etc.

Leihui Chen is currently a master in the School of Data Science and Engineering of East China Normal University, China. Her research interests include recommender systems, computational advertising , social network analysis and mining. She is a student member of CCF.

Aoying Zhou is a professor on School of Data Science and Engineering (DaSE) at East China Normal University (ECNU), where he is heading DaSE and vice president of ECNU. He is the winner of the National Science Fund for Distinguished Young Scholars supported by NSFC and the professorship appointment under Changjiang Scholars Program of Ministry of Education. His research interests include Web data management, data intensive computing, in-memory cluster computing and benchmark for big data. His works appear in major international journals and conferences, including TKDE, PVLDB, SIGMOD, SIGIR, KDD, WWW, ICDE and ICDM, etc.