World Wide Web (2023) 26:3463–3481
https://doi.org/10.1007/s11280-023-01189-5

ASBiNE: Dynamic Bipartite Network Embedding for incorporating structural and attribute information

Sajjad Athar[1] · Rabeeh Ayaz Abbasi[1] · Zafar Saeed[2] · Anwar Said[3] · Imran Razzak[4] · Flora D. Salim[4]

Received: 4 March 2023 / Revised: 18 May 2023 / Accepted: 16 June 2023 /
Published online: 28 July 2023

Abstract
Graph representation learning (GRL) has recently gained attention and becoming popular in research community. GRL has been proven to be extremely handy for transforming large and complex network data onto a low-dimensional vector space. It opens the door to many vector-based algorithms such as link prediction, recommendation, and classification to be effectively applied to the network data. Though many GRL algorithms exist for homogeneous (one-mode) graphs, however only a few methods exist for bipartite (two-mode) graphs. Most of the existing methods for bipartite graph representation learning mainly focus on graph topology and ignore the information available in the attributes of the nodes. In this paper, we propose a novel "Attributed and Structural Bipartite Network Embedding" (ASBiNE) method. The ASBiNE incorporates both the graph topological information concerning inter-partition and intra-partition links and attributes information by generating proximity between nodes having attribute similarities. Intermediate embeddings are generated by modeling the intra-partition links in homogeneous structural and attribute networks separately, which in the end are combined through a joint optimization framework, and final embeddings are generated. The attribute and structural information share is controlled before the joint optimization step. The proposed method is evaluated on a real-life dataset through extensive experiments. The results show that the proposed method is effective and outperforms state-of-the-art baseline embedding methods.

Keywords Bipartite networks · Graph representation learning · Neural networks · Recommendation · graphSAGE

This article belongs to the Topical Collection: *Special Issue on Fairness-driven User Behavioral Modelling and Analysis for Online Recommendation*
Guest Editors: Jianxin Li, Guandong Xu, Xiang Ren, and Qing Li.

✉ Rabeeh Ayaz Abbasi
rabbasi@qau.edu.pk

Extended author information available on the last page of the article

1 Introduction

Modern-day technology allows the development of complex systems such as systems dealing with social interactions (social networks), e-commerce systems, biological systems, and recommendation systems. To deal with the inherent complexity of these real-life systems, they can be designed as graphs or networks to reflect interlinked information [1]. Networks can be an elegant choice to model the interacting entities or objects in these systems. Networks store the information in the form of nodes or vertices which represent the peoples as in the case of social networks and edges represent the interactions or relationships between them.

Traditionally, networks are represented by high dimensional adjacency matrices, thus having sparse representations. The vertices in these matrices are represented by row vectors of the order of nodes N in the network. Data analytics using traditional graph representations has become a challenging task due to the enormous network size with the ever-increasing data in real-life systems. This computational bottleneck in sparse networks motivates the research community to devise useful graph representations with reduced dimensions while preserving structural information. This is where the network embedding field comes under the limelight and became an efficient approach to represent the modern-day huge networks.

Network embedding methods map the networks onto a low dimensional continuous vector space and has been proven to have significantly enhanced the performance and efficiency of deep learning tasks such as node classification, link prediction, recommendation [2, 3], etc. Since the last decade, many novel embedding methods have been proposed, most dedicated to homogeneous networks [4–6]. Nevertheless, the systems that we come across in our daily life are primarily heterogeneous in nature. The most common example of heterogeneous networks is bipartite networks [7]. In recent years, these complex networks, especially bipartite networks, have drawn the attention of researchers. Most of the bipartite embedding methods [8–14] exploit only the structural information. A few methods [11, 15] use node features reflecting empirical characteristics often attached to the real-life networks.

This paper presents an attributed bipartite embedding method that incorporates the structural and feature information of the vertices. The proposed method dynamically adjusts the vertices' structural and attribute information into the final embeddings depending on the requirement of the machine learning task. A Bipartite network is a special type of network containing two partitions or sets of vertices. An edge is allowed between two vertices only if they belong to different sets. These are called explicit or observed edges that capture the inter-partition proximities between the distinct types of vertices. In real-life networks, a significant portion of network structure [6] is contained by the edges which are not observed, these are intra-partition edges or implicit edges [10, 16] which capture the proximities between the vertices within each partition. If we only consider the explicit edges for preserving the network's structure, we could lose important semantic information contained by these implicit links. If two vertices u_k and u_{k+1} from a disjoint partition U have a common neighbor v_m in partition V then u_k and u_{k+1} have a semantic relationship. For instance, consider a real-life scenario from an e-commerce application to understand the importance of such a relationship. If two customers purchase the same item, there exists a link expressing, these customers are implicitly related. This implicit relation gets stronger if the list of commonly purchased items increases. Including this information encoded into these implicit interactions between the same type of vertices can positively impact the quality of embedding vectors and eventually the machine learning tasks. The existing bipartite embedding methods also

exploit the intra-partition proximities between the vertices within each partition to generate the quality embeddings [10, 11, 15]. In case of attributed bipartite networks (each node carries auxiliary information in the form of textual attributes), intra-partition proximities also include attribute aspect of nodes. The attributed bipartite networks can model many real-world systems, e.g., the recommender system in [17], where people share their experiences and reviews about the products (hotels). The vertices are categorized into visitors and hotels, while the edges denote the visitors' experiences concerning their stay at hotels.

The most recent work on embedding attributed bipartite networks try to correlate the highly non-linear relationship between structural and attribute proximities by treating them as two different information modalities and modeling them separately [11]. They maximized the structural and attribute information correlation between the nodes connected in the intra-partition network, arguing that attribute complementarity exists between structurally connected nodes. In our research study, we are interested in exploring nodes that are structurally located at disparate places but connected on the attribute space. Sheikh et al. [18] highlighted that the attributes information becomes extremely useful in producing quality embeddings when vertices show high feature similarities but topologically do not lie in close vicinity. We also perform the intra-partition proximity modeling for preserving structural and attribute information, independently and construct the structural and attribute intermediate embeddings.

Our proposed method can dynamically proportionate the structural and attribute information into the final representations as continuous vector space depending on the nature of the application. The motivation behind this research is to design an embedding method that should be able to generate representations based on the nature of the machine-learning task. There could be situations when vertices' attribute aspect is more important than their structural aspect, for instance, a talent hunt program to find some promising players belonging to different regions, for a certain sport. In this case, the players' attributes are more critical than their topological placements in the network. Therefore, changing the values of hyperparameters controlling the information shares changes the attribute and structural neighborhoods of the vertices. graphSAGE implements neighbor sampling and aggregation to generate the embeddings, while pinSAGE, a variant of graphSAGE defines the neighborhood samples based on their importance. The importance of a neighbor is measured as the number of its appearances on the random walks started from a node. ASBiNE: utilizes the structural as well as attribute information, it differs from graphSAGE in that it could bring the structurally distant nodes due to their attribute similarities closer on the final embedding space. We can also prioritize the structural and attribute information in the final embeddings depending on the application requirement. Key contributions of our study include:

1. An attributed bipartite network's embedding method that brings the vertices closer on the embedding space based on their attribute similarities despite the absence of structural correlations.
2. The structural and attribute modalities are modeled separately and independently of each other, giving the embedding method more freedom to become applicable to different kinds of application domains.
3. The method dynamically proportionates attribute and structural information shares into the final embeddings.
4. Empirical evaluation against seven well-known baseline methods using a publicly available benchmark dataset.

2 Related work

This section reviews the developments in the field of network embedding chronologically and critically discusses the shortcoming in the most prominent network embedding methods. We start by discussing some common embedding techniques and then how these techniques are adapted to suit specific types of networks such as homogeneous and bipartite networks.

Dimensionality reduction methods. The early instigation of graph representation learning focused on devising low dimensional representations for the network data. The existing embedding methods propose dimensionality reduction with several techniques such as Principal Component analysis (PCA), Multidimensional scaling (MDS), IsoMap, Local Linear Embeddings(LLE), and Laplacian Eigenmaps (LE) [19, 20].

Matrix factorization-based methods. The network is normally represented by an adjacency or laplacian matrix. This mathematical representation opens up an opportunity for applying mathematical operations like matrix factorization (MF) using Singular Value Decomposition (SVD) and Eigen Decomposition. Several embedding methods have been designed based on these MF techniques, for instance, GraRep [21] and HOPE [22].

Random walk-based methods. Next in the line are random walk-based embedding methods, the purpose of these methods is to preserve the network structure. The pioneer works in this category DeepWalk [4] and node2vec [5]. These methods employ Skip-gram [23] technique to learn vertex representations. The overall embedding method starts by converting the input network into a corpus of vertex sequences by performing truncated random walks, and then skip-gram is used to learn the vertex representations. Metapath2vec [24] and HIN2vec [25] are the extensions of DeepWalk, which perform random walks on heterogeneous networks.

Neural network-based network embeddings. Moreover, the deep neural network-based embedding methods, which include: 1) GCN [26] is a semi-supervised learning-based approach accepts networks with variable size and shape. 2) Variational Graph Auto-Encoders (VGAE) [27], an unsupervised framework that uses a GCN encoder and the decoder, implements the inner product functionality. The model can incorporate the feature information during training and achieve significant improvement on benchmark dataset. 3) graphSAGE [28], an inductive node embedding approach, utilizes the textual contents of the vertices to generate the embeddings for the unseen nodes added later to the network. Previously the existing transductive approaches require all the vertices to be present during training and cannot be generalized for the unknown nodes added later to the network. graphSAGE implements neighbor sampling and aggregation to generate the embeddings in the first phase, and in the second phase, a graph-based unsupervised loss is applied. 4) pinSAGE [29] is a variant of graphSAGE, the only difference lies in the generation of neighbor samples. pinSAGE defines the neighborhood samples based on their importance. The importance of a neighbor is measured as the number of its appearances on the random walks started from a node. 5) Structural Deep Network Embedding (SDNE) [30] preserves the network structure by considering 1^{st} and 2^{nd} order nodes' proximities to capture the local and global structures of the network. Some worth mentioning graph embedding methods are temporal knowledge graph via hyperbolic embedding [31] and deep-learning based knowledge tracing techniques, such as Deep Knowledge Tracing model (DKT) [32].

The embedding approaches mentioned above are all specific to homogeneous networks. The advanced embedding methods for hybrid type of networks are built on these general

embedding approaches. Because, ASBiNE method we are going to propose in this paper is about finding embeddings for attributed bipartite networks, I feel it necessary to justify why heterogeneous network approaches are found to be sub-optimal for bipartite graphs.

Heterogeneous network embeddings. Heterogeneous networks are composed of different types of vertices and/or edges, this diversity in node and edge types is the major challenge for the researchers, to embed these networks. Some most significant heterogeneous network's embedding methods are metapath2vec [24], Heterogeneous Network Embedding (HNE) [33], Relation Structure-aware Heterogeneous Information Network Embedding model (RHINE) [34], Active Heterogeneous Network Embedding framework (ActiveHNE) [35] and Embedding of Embedding (EOE) [36] etc. All the embedding methods described above (except heterogeneous techniques) are homogeneous network embedding methods. They might be a sub-optimal choice for bipartite networks because they cannot distinguish between different types of vertices. While heterogeneous embedding methods [34, 38], for the sake of argument, could be thought of a good choice for bipartite networks by considering the bipartite networks a special case of heterogeneous networks. But the way they are designed, they treat the explicit and implicit links equally and do not exploit the semantics encoded into the implicit links.

Bipartite network embeddings. BiNE [10] and BiANE [11] are the two significant methods proposed for embedding the bipartite networks. BiNE constructs the embedding vectors by modeling the implicit and explicit links. Exponential growth in computational complexity is one of the major challenges in implicit edge modeling. Therefore, biased random walks [5, 10] are performed, and skip-gram model [37] is applied, maximizing the vertices' co-occurring probabilities. The random walks are kept biased so that the vertices' visitation counts follow the power law pattern or long-tail distribution. Keeping the number of random walks from each vertex same and length equal may undermine the influence of vertices with high degrees and over-emphasize the ones with low degrees. During explicit edge modeling, a bipartite network is reconstructed from the explicit links. BiNE does not incorporate the vertices' attribute information. BiANE [11] is the embedding framework designed for attributed bipartite networks. The novel aspect of this method is the introduction of structural and attribute proximities correlation training approach by using autoencoders. The intermediate training vectors produced by the auto-encoders are fed to the joint optimization module for modeling the inter-partition proximities in the bipartite network. It also implements dynamic positive sampling to keep the computation cost minimum. PBiNE [38] extends BiNE by applying Kalman filter. The ASBiNE method is immediately based on BiANE [11] but the novelty of ASBiNE is that it gives us more command on controlling the contribution of structural and attribute information into the final embeddings.

3 Methodolody

In this section, we present the proposed framework for learning representations of attributed bipartite networks by exploiting the networks' structural and attribute information. We initially introduce some key concepts and then present the methodology of the proposed framework in the rest of the section.

3.1 Notations and problem formulation

Attributed bipartite network. The attributed bipartite network is denoted by $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \mathcal{X}_{\mathcal{U}}, \mathcal{X}_{\mathcal{V}})$, where $\mathcal{U}$ and $\mathcal{V}$ represent the two disjoint partitions or sets of vertices u_i and v_i such that $u_i \in \mathcal{U}$ and $v_i \in \mathcal{V}$. $\mathcal{E} \subseteq \mathcal{U} \times \mathcal{V}$ denotes the set of inter-partition edges. v_i $(i = 1, 2, \cdots, n)$ and v_j $(j = 1, 2, \cdots, m)$ denote the vertices in $\mathcal{U}$ and $\mathcal{V}$ where there are no intra-connectivity in both the sets. We consider the edges to be of unit weight if weights are not mentioned explicitly. We use bi-adjacency matrix $\mathrm{W} = [w_{ij}]$ of dimension $|\mathcal{U}| \times |\mathcal{V}|$ to represent the structure of bipartite network. $\mathcal{X}_{\mathcal{U}}$ and $\mathcal{X}_{\mathcal{V}}$ are the attribute matrices for the vertices in $\mathcal{U}$ and $\mathcal{V}$ (rows in the matrix represent the feature vectors for vertices). We show an example of an attributed bipartite graph in Figure 1.

Explicit edges. We denote the actual edges among the two sets $\mathcal{U}$ and $\mathcal{V}$ as explicit edges.

Implicit edges. In addition to the explicit edges, we define implicit edges in the same set, e.g., if two nodes of the set $\mathcal{U}$ share neighbors in the other set $\mathcal{V}$, the two sets in the set $\mathcal{U}$ are implicitly connected. The number of common neighbors defines the weight of the implicit edge.

Structural proximity. is interpreted as the local and global structural similarities between the vertices. The local network structure is represented by the observed or explicit edges, i.e., the vertices having a direct edge share first-order proximities or related locally on the network's structure. While global network structure is represented by the unobserved or

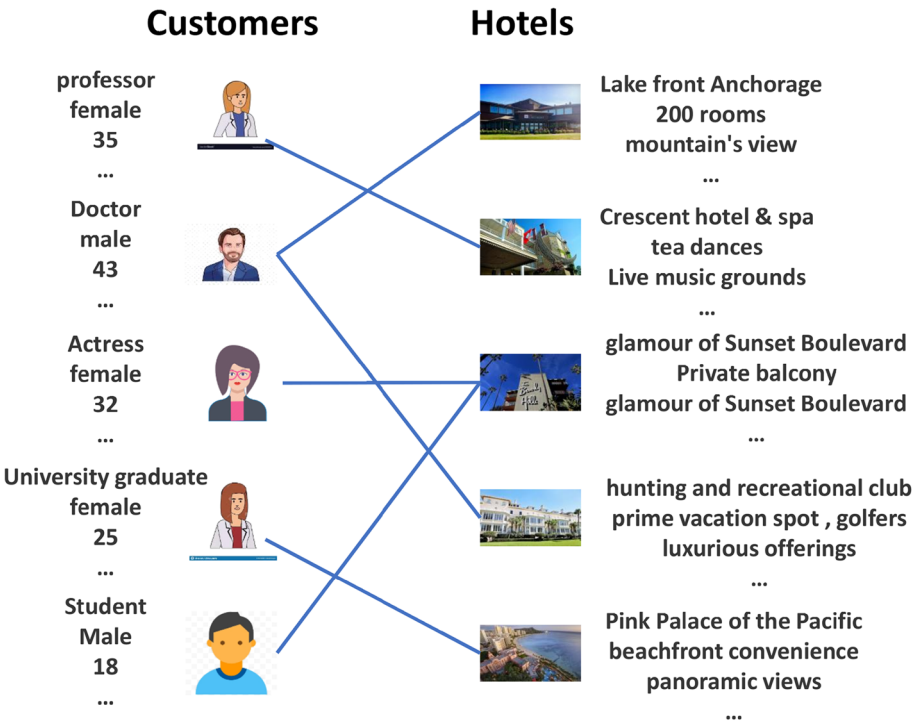

Figure 1 Attributed bipartite network: hotel management system

implicit edges which capture the high-order proximities between the vertices, i.e., vertices having neighborhood similarities.

Attribute proximity. is the measure of similarity between the attributes of pairs of nodes in an attributed bipartite network. By incorporating the attribute information in the embedding space, it is more likely that the vertices with similar attributes are placed closer to each other in the embedding space [11].

3.2 ASBiNE: attributed and structural Bipartite Network Embedding

The task of learning the structure of bipartite networks is quite challenging as compared to homogeneous networks. One of the main reasons could be that diverse types of nodes and multi-modal attributes are involved, limiting the applicability of the simple graph representation approaches [8, 10, 16].

ASBiNE generates representations by preserving the network's structural as well as attribute proximity information. Considering the significance of attribute and structural information, we perform explicit and implicit edge modeling. While the attribute information is preserved by modeling the attribute similarities between vertices of the same type. In the end we cloud prioritize the structural and attribute information in the final embeddings. The embedding method combines the two aspects of the bipartite network by following the properties of consistency and complementarity [39, 40]. Consistency tends to maximize the similarities among different aspects of the data, while complementarity states that the network's aspects are disjointed, i.e., one aspect may contain information others may not have. In this context, the vertices hold attribute and structural information. The proposed method combines both aspects to improve the quality of embeddings. As a result, similar vertices will lie closer to each other in the final embedding space, hence, it follows consistency. Since, for vertices having disparate and disjointed attribute and structural information modalities, the proposed method combines them into the final embeddings, hence reflecting the complementarity property.

The embedding process starts by partitioning the bipartite network into homogeneous networks, i.e., projection and attribute homogeneous networks. The projection homogeneous networks are merely the bipartite network's one-mode projections while the attribute networks are generated by connecting nodes based on their attribute similarities. Edges in both these networks are modeled separately for generating intermediate embeddings, i.e., each node will be having intermediate structural and attribute embeddings. In the next step, final embeddings are generated by modeling the explicit links which exist between the nodes belonging to different partitions. During this step, we also implement our novel functionality of adjusting the feature and structural information in the final embeddings. The overall embedding process is shown in Figure 2. Formally, the whole process is described in the below sections.

3.2.1 Modelling implicit relations

The process initiates from modeling intra-partition modalities contained in the implicit edges. We partition the attributed bipartite network into four homogeneous networks, two structural $\mathcal{G}_z^u(\mathcal{U}, \mathcal{E}_z)$, $\mathcal{G}_z^v(\mathcal{V}, \mathcal{E}_z)$ and two attributed $\mathcal{G}_x^u(\mathcal{U}, \mathcal{E}_x)$, $\mathcal{G}_x^v(\mathcal{V}, \mathcal{E}_x)$ networks for both sets of vertices: while $\mathcal{E}_z$ and $\mathcal{E}_x$ represent the two sets containing structural and attribute edges.

The structural homogeneous networks (bipartite network's one mode projections) are generated from the bipartite network by connecting pairs of nodes within each partition,

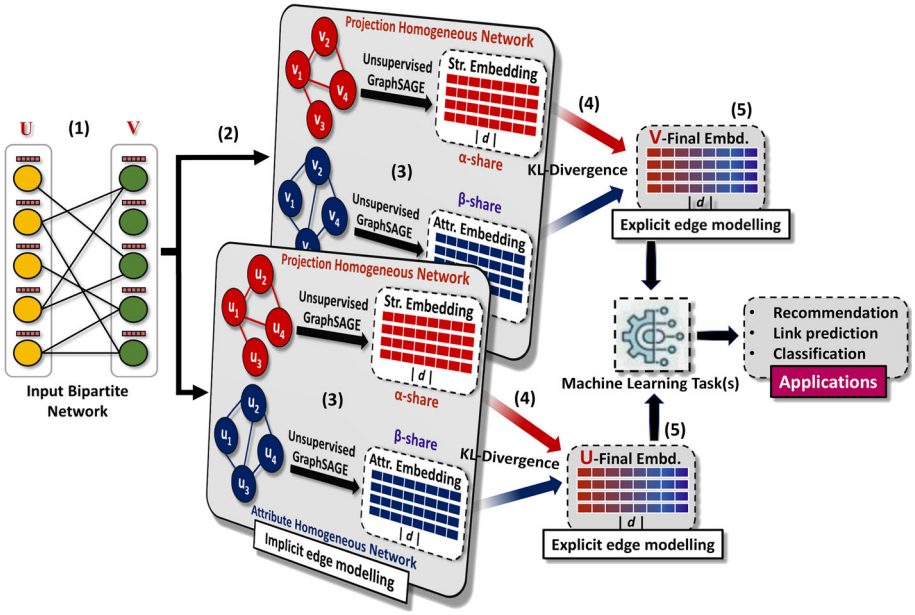

Figure 2 ASBiNE process flow. (1) the embedding learning process starts by feeding a bipartite attributed network (2) structural and attribute homogeneous networks are constructed. The intra-partition links or implicit edges are modeled using graphSAGE framework and structural and attribute intermediate embeddings are generated, (3) this is the start of explicit links modeling, this is where the structural and attribute information is adjusted, (4) explicit edge modeling using kl-Divergence, (5) the final embeddings and test bipartite network is fed to the applications module where, the embeddings are tested against different machine learning tasks: recommendation, link prediction, and classification

such that there exists a path of length 2 between them. The existence of path between the pairs of nodes provides evidence for some sort of implicit interaction between them. In other words, the nodes from one set in bipartite network having common neighbors will be connected in the structural homogeneous networks. The strength of the connection depends on the number of common neighbors. The 2nd order proximity between two vertices is defined by the following equation [10].

$$w_{ij}^{U} = \sum_{k \in V} w_{ik} w_{jk}; \quad w_{ij}^{V} = \sum_{k \in U} w_{ki} w_{kj} \tag{1}$$

The above equation computes the weights w_{ij} for the edges between the nodes in both types ($\mathcal{U}$ and $\mathcal{V}$) of structural homogeneous networks. The following matrices G_z^u and G_z^v represent the structural homogeneous networks.

$$G_z^u = [w_{ij}^{U}] \quad \text{and} \quad G_z^v = [w_{ij}^{V}] \tag{2}$$

The intra-partition proximities between the nodes within the same partition contain both the structural and attribute information aspects. The implicit relations in the attribute homogeneous networks are modeled to preserve the attribute information.

3.2.2 Preserving attribute information

To preserve the attribute information of the input bipartite network, we generate attributed homogeneous networks $\mathcal{G}_x^u(\mathcal{U}, \mathcal{E}_x)$ and $\mathcal{G}_x^v(\mathcal{V}, \mathcal{E}_x))$ for both types of vertices. Since vertices in each partition contain attribute information in the form of feature vectors, the attributed homogeneous networks are constructed by connecting vertices based on the cosine similarities of their feature vectors passing a certain threshold. $\mathcal{U}_\mathcal{A}$ and $\mathcal{V}_\mathcal{A}$.

$$[s_{ij}^U] = Sim(u_i, u_j) \quad \text{and} \quad [s_{ij}^V] = Sim(v_i, v_j) \tag{3}$$

The following matrices represent the attribute homogeneous networks.

$$G_x^u = [s_{ij}^U] \quad \text{and} \quad G_x^v = [s_{ij}^V] \tag{4}$$

3.2.3 Unsupervised-GraphSAGE

We use the graphSAGE [28] framework to learn intermediate structural and attribute embeddings on these homogeneous networks in an unsupervised way. Because we employ graph-based loss during training, we term it unsupervised graphSAGE [28]. During the feature aggregation step, graphSAGE used importance-based neighbor samples, i.e., neighbor samples are composed of nodes with top centrality values [29]. The implicit modeling is stated in the Algorithm 1. The algorithm shows training for one partition, i.e., for G_z^u and G_x^u homogeneous networks, but we repeat this process for other networks as well. We use static negative sampling to reduce the computational overhead of the algorithm [10, 16, 28]. To preserve the intra-partition proximity in the homogeneous networks, we optimize the following loss functions.

$$\mathcal{L}_1 = -\log\left(\sigma(\mathbf{Z_i^u}^\top \mathbf{Z_{n_i}^u})\right) - Q \cdot E_{n'} \sim P_{n'} \log\left(\sigma(-\mathbf{Z_i^u}^\top \mathbf{Z_{n_i'}^u})\right) \tag{5}$$

$$\mathcal{L}_2 = -\log\left(\sigma(\mathbf{X_i^u}^\top \mathbf{X_{n_i}^u})\right) - Q \cdot E_{n'} \sim P_{n'} \log\left(\sigma(-\mathbf{X_i^u}^\top \mathbf{X_{n_i'}^u})\right) \tag{6}$$

$$\mathcal{L}_3 = -\log\left(\sigma(\mathbf{Z_i^v}^\top \mathbf{Z_{n_i}^v})\right) - Q \cdot E_{n'} \sim P_{n'} \log\left(\sigma(-\mathbf{Z_i^v}^\top \mathbf{Z_{n_i'}^v})\right) \tag{7}$$

$$\mathcal{L}_4 = -\log\left(\sigma(\mathbf{X_i^v}^\top \mathbf{X_{n_i}^v})\right) - Q \cdot E_{n'} \sim P_{n'} \log\left(\sigma(-\mathbf{X_i^v}^\top \mathbf{X_{n_i'}^v})\right) \tag{8}$$

Z^u, Z^v and X^u, X^v are the input random vectors for nodes $\mathcal{U}$ and $\mathcal{V}$ respectively, fed during optimization. Q denotes the size of the negative sample. n_i' are the negative sample nodes, i.e., the node lying at distant locations from the node in focus, while $\mathbf{n_i}$ are the neighboring nodes and comprise the positive sample nodes. $P_{n'}$ represents the distributions of negative samples. At the end of this phase, we can construct the structural and attribute intermediate embeddings, i.e., $\vec{z^u}$, $\vec{x^u}$, $\vec{z^v}$ and $\vec{x^v}$ for all the nodes in sets $\mathcal{V}$ and $\mathcal{U}$.

3.2.4 Modelling explicit relations

We follow the same scheme for modeling the explicit edges as in papers [6, 10, 16]. The explicit links are modeled by maximizing the vertex co-occurring probabilities between the nodes connected through explicit links, in the network and in the embedding space. The vertex co-occurring probability between vertices u_i and v_j in the network space is:

$$P(i, j = \frac{w_{ij}}{\sum_{e_{ij} \in E} w_{ij}} \tag{9}$$

Algorithm 1 Unsupervised graphSAGE

Input : Homogeneous networks $\mathcal{G}_z^u(\mathcal{U}, \mathcal{E}_z)$ and $\mathcal{G}_x^u(\mathcal{U}, \mathcal{E}_x)$; Random features Z^u and X^u; no. of network layers K; Weight matrices W^k for each of the layers; positive samples $N_z(u)$, $N_x(u)$ and negative samples $N_z'(u)$, $N_x'(u)$ $\forall u \in \mathcal{U}$

respectively **Output**: structural and attribute embedding vectors $\vec{z^u}$ and $\vec{x^u}$ for all $u \in \mathcal{U}$
for $k = 1...K$ do
— $h_u^0 \leftarrow Z^u$
— for $u \in \mathcal{G}_z^u$(do
—— $h_{N_z(u)}^k \leftarrow AGGREGATE_k\left(\{h_u^{k-1}, \forall u \in N_z(u)\}\right)$
—— $h_u^k \leftarrow \sigma\left(W^k \cdot CONCAT\left(h_u^{k-1}, h_{N_z(u)}^k\right)\right)$
— end
— $h_u^0 \leftarrow X^u$
— for $u \in \mathcal{G}_x^u$(do
—— $h_{N_x(u)}^k \leftarrow AGGREGATEk_k\left(\{h_u^{k-1}, \forall u \in N_x(u)\}\right)$
—— $h_u^k \leftarrow \sigma\left(W^k \cdot CONCAT\left(h_u^{k-1}, h_{N_x(u)}^k\right)\right)$
— end
end
return $\vec{z^u}$ and $\vec{x^u}$: for all $u \in \mathcal{U}$

w_{ij} denote the weight of the edge e_{ij}. The vertex co-occurring probability between nodes u_i and v_j on the embedding space is estimated by taking the dot product between vectors $\vec{u_i}$ and $\vec{v_j}$ [23]:

$$\hat{P}(i, j) = \frac{1}{1 + exp(-\vec{u_i}^\top \vec{v_j})} \tag{10}$$

$\vec{u_i} \in R^d$ and $\vec{v_j} \in R^d$ are the d-dimensional vectors of nodes u_i and v_j. The explicit relations are preserved by minimizing the difference between these distributions (Eq. 2). We measure this difference by using KL-Divergence [6] below:

$$\begin{aligned} \mathcal{L}_5 = KL(P \parallel \hat{P}) &= \sum_{e_{ij} \in E} P(i, j) \log(\frac{P(i, j)}{\hat{P}(i, j)}) \\ &\propto -\sum_{e_{ij} \in E} w_{ij} \log(\hat{P}(i, j)) \end{aligned} \tag{11}$$

3.2.5 Joint optimization

To construct final embeddings for the Bipartite network, we jointly optimize the previously generated intermediate embeddings for homogeneous networks by combining all the optimization functions.

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 + \mathcal{L}_4 + \mathcal{L}_5 \tag{12}$$

Before performing the joint optimization, we need to combine the structural and attribute intermediate embeddings ($\vec{z^u}$, $\vec{x^u}$, $\vec{z^v}$ and $\vec{x^v}$) to create the vectors $\vec{u_i}$ and $\vec{v_j}$. This is the point where we control the participation of structural and attribute information by adjusting the values of hyperparameters and α and β, Eqs. 13 and 14.

$$\vec{u_i} = \left[(\alpha * \vec{z_i^u}) + (\beta * \vec{x_i^u})\right] \tag{13}$$

$$\vec{v_j} = \left[(\alpha * \vec{z_j^v}) + (\beta * \vec{x_j^v})\right] \tag{14}$$

The joint optimization step is elaborated in the following equations, λ is the learning rate during optimization.

$$\vec{u_i} = \vec{u_i} + \lambda\{w_{ij}\left[1 - \sigma\left(\vec{u_i}^\top \vec{v_j}\right)\right] \cdot \vec{v_j}\} \tag{15}$$

$$\vec{v_j} = \vec{v_j} + \lambda\{w_{ij}\left[1 - \sigma\left(\vec{u_i}^\top \vec{v_j}\right)\right] \cdot \vec{u_i}\} \tag{16}$$

4 Complexity analysis

In order to perform the complexity analysis, we divide the whole process of generating representation vectors ($\vec{u_i}$ and $\vec{v_j}$) into small steps and accumulate the cost of each step.

Structural homogeneous networks. To generate structural homogeneous networks (G_z^u and G_z^v) we perform dot-product between each type of similar nodes, i.e., $\vec{u_i}^\top \vec{u_j}$ and $\vec{v_i}^\top \vec{v_j}$ for $i \neq j$. The dimensionalities of vertices $\mathcal{U}$ and $\mathcal{V}$ are $n \times n$ and $m \times m$ respectively, n being the number of U nodes (n = $|\mathcal{U}|$) and m being the number of V nodes (m = $|\mathcal{V}|$). The running time for both the structural homogeneous networks is $\max(n^2, m^2)$, formally $O\left((\max(|\mathcal{U}|, |\mathcal{V}|))^2\right)$.

Attribute homogeneous networks. To generate structural homogeneous networks (G_z^u and G_z^v) we perform dot-product between each type of similar nodes, i.e., $Sim(u_i, u_j)$ and $Sim(v_i, u_j)$ for $i \neq j$. The time complexity for generating attribute homogeneous networks is same as the structural homogeneous networks except dot product operation is replaced by cosine similarity, i.e., $O\left((\max(|\mathcal{U}|, |\mathcal{V}|))^2\right)$

GraphSAGE (intermediate embeddings). We use graphSAGE framework to generate intermediate structural and attribute embeddings. We used fixed size neighbor samples instead of using all neighbors ($N_z(u)$, $N_x(u)$) and ($N_z(v)$, $N_x(v)$). The expected run time would be unpredictable without fixed-size neighbor samples. The running time for a graphSAGE layer is $O\left(r^k\,|\mathcal{U}|\,d^2\right)$ or $O\left(r^k\,|\mathcal{V}|\,d^2\right)$ [28, 41] for the vertices $\mathcal{U}$ and $\mathcal{V}$, where k is the number of layers, n is the total number of nodes, r is the number of sampled neighbors and d is the dimension of intermediate embedding and is a lot less than $\mathcal{U}$ or $\mathcal{V}$ (d $\ll \mathcal{U}$ or d $\ll \mathcal{V}$). Simply the time complexity of graphSAGE for both structural and homogeneous networks is $2(O\left(|\mathcal{U}|\right))$ and $2(O\left(|\mathcal{V}|\right))$

KL-divergence (joint optimization). The joint optimization framework employing KL-divergence, the number of operations goes to the order of the number of edges $|\mathcal{E}|$. In worst case the the number of edges reaches to $m \times n$, so the time complexity of this joint optimization step is $O\left((\max(|\mathcal{U}|, |\mathcal{V}|))^2\right)$

The cumulative time complexity for the ASBiNE method is:

$$\begin{aligned} Complexity(ASBiNE) = {} & O\left((\max(|\mathcal{U}|, |\mathcal{V}|))^2\right) + O\left((\max(|\mathcal{U}|, |\mathcal{V}|))^2\right) \\ & + 2(O\left(|\mathcal{U}|\right)) + 2(O\left(|\mathcal{V}|\right)) + O\left((\max(|\mathcal{U}|, |\mathcal{V}|))^2\right) \end{aligned} \tag{17}$$

5 Experiments and results

This section describes the experimental setup and details of the real-world benchmark dataset. The performance of the proposed ASBiNE method is evaluated by comparing the results

Table 1 Movielens dataset statistics

Dataset	#Users $\|\mathcal{U}\|$	#Items$\|\mathcal{V}\|$	#Links	Attributes$\|\mathcal{U}\|$	Attributes$\|\mathcal{V}\|$
MovieLens	943	1682	100,000	21	18

with seven well-known baseline network embedding methods. The section also discusses the evaluation criteria and hyper-parameters, followed by results and discussion.

5.1 Dataset

Movielens: [1] is a publicly available dataset that contains 100,000 anonymous users' ratings (**ml_100k**) with 943 users and 1682 movies. At least 20 ratings have been recorded for each user. The users and movies also carry attribute information. We consider three attributes, gender (male or female), age (normalized age values within the range of 0 to 1), and occupation (list of 21 unique occupations) for the users. While for movies, we consider movie *genres* having 18 categorical values. To generate the feature vectors for users and movies, we first generate one-hot encoding [42] for the categorical attributes and then concatenate them. Table 1 provides the detail about the movielens dataset, while the users' and movies' attribute vectors are explained in Figures 3 and 4.

5.2 Baselines

1. **DeepWalk** [4] algorithm generates a corpus of node sequences by performing fixed-length, uniform random walks for each vertex in the network. Then skip-gram model is applied on the corpus to learn embeddings for the homogeneous network.
2. **node2vec** [5] this approach is based on deepWalk. Unlike deepWalk, it introduces biasedness in generating the vertex sequences by using hyperparameters p and q. The vertex sampling strategy either follows depth-first or breadth-first search pattern. Setting p and q value equal to 0.5 produces good results.
3. **LINE** [6] this approach generates separate intermediate embeddings by modelling 1^{st} 2^{nd} proximities in the homogeneous network. The intermediate embeddings are concatenated to produce final embeddings.
4. **Metapath2vec++** [24] learns embeddings for multiple types of nodes in a heterogeneous network. It generates a corpus of vertex sequences by performing metapath-based random walks. Then Skipgram algorithm is employed to model the node's interactions which are topologically and semantically related.
5. **BiNE** [10] is the main baseline method, we have built our method on. It learns the representation of the bipartite networks by modeling the observed inter-partition edges and intra-partition links which capture the 2^{nd} proximities between the nodes of the same type. The representation is based on only structural similarities and does not include the attributes.
6. **ABiNE** [15] learns the latent representations for attributed bipartite networks. This method also exploits the structural and attribute information in the context of implicit and explicit relations.

[1] https://grouplens.org/datasets/movielens/

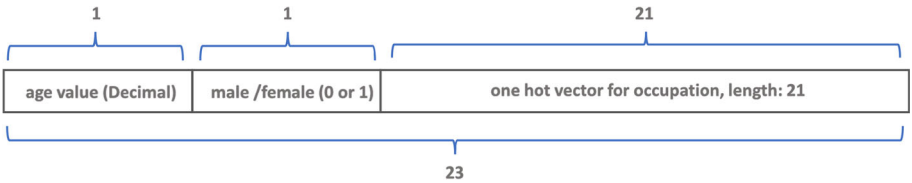

Figure 3 Format of users' feature vectors, the lenght of the vector is 23

7. **BiANE** [11] learns the embeddings for attributed bipartite networks. It models the implicit and explicit interactions between the vertices, and also incorporates attributes information. Embeddings are generated through auto-encoder-based training.

5.3 Evaluation criteria

Recommendation: is a ranking problem that gives top-ranked items for each user according to their preferences or interests. we use the Movielens dataset and follow the same scheme adopted in BiNE [10]. We construct the training set by randomly sampling 70% edges of the Movielens dataset and the remaining 30% edges used as the test set. We get the top 10 ranked movies for each user and evaluate the performance with four widely used measures F1, Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR) [43].

5.4 Hyperparameter settings

To test all the baselines, we tuned the hyperparameters according to their default implementations producing the best results. The values of these parameters impact the performance of the embedding methods. Unlike BiNE [10] we have performed the implicit and explicit links training separately and independently. For Explicit link and joint optimization, we follow the BiNIE implementation. The original idea for explicit link optimization floated from LINE [6].

We kept the embedding vector size to 128 as in the baseline [10]. The hyperparameters **pos_exam** and **neg_exam** are used by graphSAGE during implicit edge modeling. Sample size of 30 and 50 nodes for **pos_exam** and **neg_exam** parameters respectively empirically proved to be the good choices. **n_layers**: We use single-layer graphSAGE (with important neighbor sampling) on the homogeneous networks during implicit link training. Since the edges capture the 2^{nd} order proximities between the vertices, therefore we don't need to have two convolution layers in the graphSAGE. To model the 2^{nd} order relationships only one-layer graphSAGE would do the job. **l_rate**: we tried [0.5, 1, 1.5, 2, 2.5, 3, 5] values for **l_rate**, 2.0 is found to be the best choice. We set **neig_samp** value as 30. **attr_thresh**: this value controls the number of attribute edges. We set 0.7 for **attr_thresh** which best tradeoff

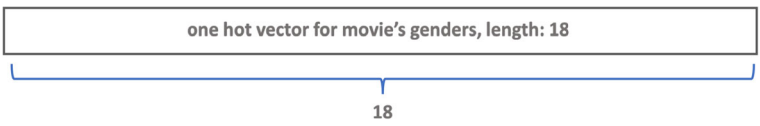

Figure 4 Format of movies' feature, total length is 18

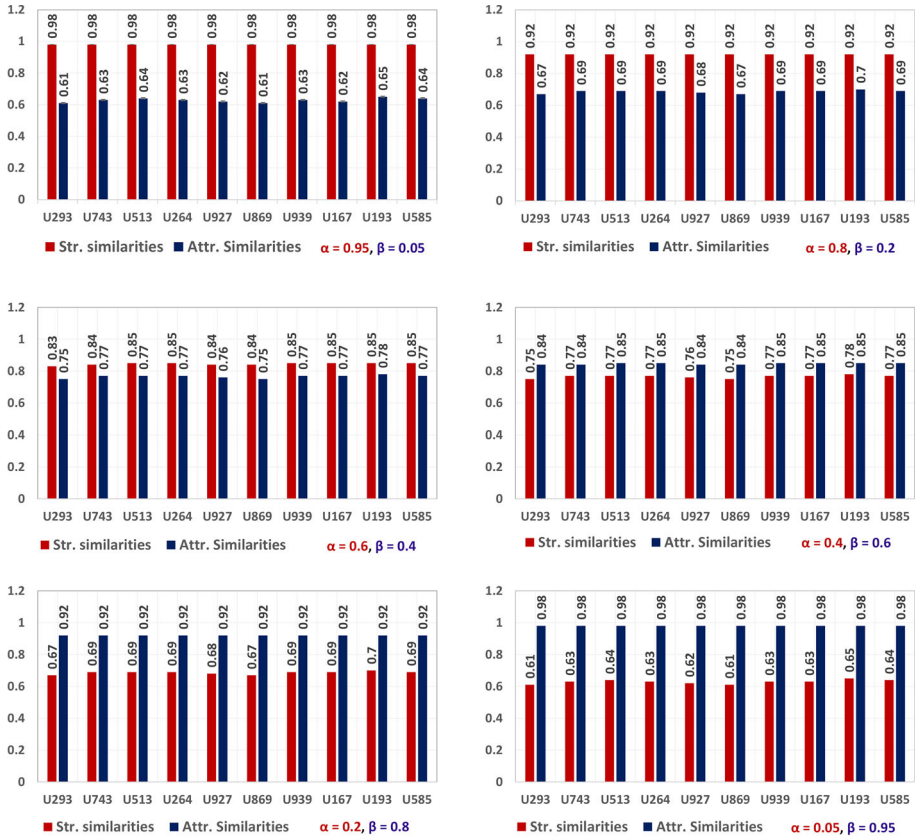

Figure 5 Testing the structural and attribute information contribution: we randomly select 10 nodes adjust the participation of attribute and structural information into the final embedding before joint optimization

between accuracy and computation complexity of our model. We fixed the value of γ trade-ff parameters at 0.1, as in **BiNE**. In the end, the learning rate for explicit link training λ is set to be 0.25. Table 2 provides the details about the hyperparameters during the experiments.

5.5 Discussion

We get the top 10 most preferred or favorite movies for each user. Preference is estimated by taking dot product $u_i^\top v_j$ between the user's embedding vector u_i and movie's embedding vector v_j learned by our method. **ASBiNE** outperforms the baseline methods. The most prominent among them are **BiNE**, **metapath2vec**, and **LINE** which utilize the implicit edges or 2^{nd} order proximities between the vertices. If we discuss one by one: in case of **BiNE** the reason is adding the attribute links which compensate for most of the missing links which is a normal case in real-life networks [44]. Another reason is that the BiNE performs importance-based random walks on the vertices to seek context nodes or positive examples. The length of walks increases for important nodes and the strategy becomes biased towards depth-first, but we include a sample of fixed-length neighbors or positive nodes based on their importance.

Table 2 Range of values for hyperparameters used in **ASBiNE**: optimal values are in bold underline

Parameters	Meaning	Experiment Values
pos_exam	number of positive examples	[**30**, 40, 50, 60, 70, 100]
neg_exam	number of negative examples	[30, 40, **50**, 60, 70, 100]
n_layers	number of graphSAGE layers	[**1**, 2]
l_rate	learning rate: implicit link training	[0.5, 1, 1.5, **2**, 2.5, 3, 5]
neig_samp	graphSAGE: neighbor sample size	[**30**, 50, 70, 100]
attr_thresh	cosine similarity threshold for attribute network	[0.5, 0.6, **0.7**, 0.8, 0.9]
γ	trade-off parameter: explicit link training	[0.0001, 0.001, 0.01, **0.1**, 1]
λ	learning rate for explicit link optimization	[0.01, 0.05, 0.1, **0.25**, 0.5, 1, 5]
α	hyperparameter controlling structural information	[0.05, 0.2, 0.4, **0.6**, 0.8, 0.95]
β	hyperparameter controlling attribute information	[0.05, 0.2, **0.4**, 0.6, 0.8, 0.95]

metapath2vec also incorporates the implicit links but the reason for inconsistency in results is that it does not distinguish between implicit and explicit edge weights and treats them equally.

LINE though preserves both the implicit and explicit relations but the reason it lags significantly behind our model is that it does not have any joint framework to unify the intermediate implicit and explicit embeddings.

The **ASBiNE** besides preserving the implicit and explicit relations also adds the attribute links to compensate for the missing links in networks. Including attribute improves in results, but an issue raised in **BiANE** [11] the attribute and structural edges must be correlated before feeding to the training model, we kept the attribute edges and structural edges independent of each other and perform the edge modeling separately and independently. Table 3 shows the results for recommendation tasks for **ASBiNE** and the baseline methods.

We found the old evaluation setup defined in the method [11] insufficient in our case. In baseline, they only maximize the structural and attribute information correlation between the nodes already connected through implicit links. In our case, we bring the nodes distantly placed on the structural space but having strong attribute proximities, because these nodes could be of great importance [18] depending on the nature of the application.

Table 3 Recommendation task Performance comparison for Top-10 nodes on MovieLens dataset

Algorithms	Movielens			
	F1@10	NDCG@10	MAP@10	MAR@10
DeepWalk	3.73%	3.21%	0.90%	15.40%
Node2vec	4.15%	3.68%	1.05%	18.34%
LINE	6.91%	6.51%	1.74%	20.43%
Metapath2vec	4.65%	4.39%	1.91%	16.62%
BiNE	8.05%	7.41%	2.94%	21.17%
ASBiNE	**13.49%**	**12.79%**	**4.16%**	**29.98%**

To verify, we select ten nodes from each of the partitions randomly with their intermediate structural and art tribute embeddings, i.e., $\vec{z_i^u}$, $\vec{x_i^u}$, $\vec{z_j^v}$ and $\vec{x_j^v}$. We combine these vectors in different proportions determined by α and β hyperparameters and generate resultant embedding vectors $\vec{u_i}$ and $\vec{v_i}$. We plot the spatial distances of the resultant embeddings from their structural and attribute counterparts in Figure 5. The intermediate embedding vectors which we fed at the explicit modeling step, carry the structural and attribute information shares that we have decided through α and β hyperparameters.

6 Conclusion

This paper presents **ASBiNE** method that learns the representations for bipartite attributed networks' vertices. The proposed method dynamically characterizes the attribute and structural information shares into a continuous vector space representation. The focus of this study was to devise a novel method that brings the nodes closer in the embedding space concerning their attribute similarity even if the nodes have less structural connectivity. We observed that embedding quality deteriorates when high-order relations beyond the second proximities are included, especially when the dataset is dense. Nevertheless, we must find a way to preserve this valuable information encoded in the high-order relations in the network. We include the attribute information through attribute homogeneous networks and generate the intermediate embeddings.

Lastly, we found the existing evaluation setup used in the baselines is insufficient. Therefore, we design a new evaluation setup to assess the impact of adjusting the information shares on the embedding space. The detailed experiments on a real-world dataset show that the proposed method ASBiNE has significant results; thus, the method can effectively be used for a variety of real-life problems related to classification, prediction, and recommendation.

Author contributions Sajjad Athar performed experimental analysis and design, Rabeeh Ayaz Abbasi led the experiment. Zafar Saeed and Anwar Said assisted in writing and experiments. Imran Razzak and Flora Salim assisted in methodology and writing.

Funding Zafar Saeed was funded by the project FAIR - Future AI Research (PE00000013), Spoke 6 - Symbiotic AI, under the NRRP MUR program funded by NextGenerationEU.

Availability of data and material All datasets used in this study are publicly available.

Declarations

Ethical approval This research does not involve both human and/ or animal studies.

Conflict of interest The authors declare no known potential conflict of interest.

References

1. Jia, Y., Gu, Z., Jiang, Z., Gao, C., Yang, J.: Persistent graph stream summarization for real-time graph analytics. World Wide Web 1–21 (2023)
2. Arsov, N., Mirceva, G.: Network embedding: an overview. Preprint at http://arxiv.org/abs/1911.11726 (2019)

3. Wang, Y., Yao, Y., Tong, H., Feng, X., Jian, L.: A brief review of network embedding. Big Data Min. Anal. **2**(1), 35–47 (2019)
4. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA, pp 701–710. Association for Computing Machinery (2014)
5. Grover, A., Leskovec, J.: Node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, New York, NY, USA, pp 855–864. Association for Computing Machinery (2016)
6. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, WWW '15, Republic and Canton of Geneva, CHE, pp 1067–1077. International World Wide Web Conferences Steering Committee (2015)
7. Spanurattana, S., Murata T.: Visual analysis of bipartite networks. In: 2013 IEEE 13th International Conference on Data Mining Workshops, Los Alamitos, CA, USA, pp. 833–838. IEEE Computer Society (2011)
8. Xue, H., Yang., L., Rajan, V., Jiang, W., Wei, Y., Lin, Y.: Multiplex bipartite network embedding using dual hypergraph convolutional networks. In: Proceedings of the Web Conference 2021, WWW '21, New York, NY, USA, pp 1649–1660. Association for Computing Machinery (2021)
9. Cao, J., Lin, X., Guo, S., Liu, L., Liu, T., Wang, B.: Bipartite graph embedding via mutual information maximization. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, New York, NY, USA, pp 635–643. Association for Computing Machinery (2021)
10. Gao, M., Chen, L., He, X., Zhou, A.: Bine: Bipartite network embedding. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR '18, New York, NY, USA, pp 715–724. Association for Computing Machinery (2018)
11. Huang, W., Li, Y., Fang, Y., Fan, J., Yang, H.: Biane: Bipartite attributed network embedding. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20, New York, NY, USA, pp 149–158. Association for Computing Machinery (2020)
12. Na, S., Luo, Y., Yang, Z., Wang, Z., Kolar, M.: Semiparametric nonlinear Bipartite graph representation learning with provable guarantees. Preprint at http://arxiv.org/abs/2003.01013 (2020)
13. Li, R., Zhang, S., Wan, B., He, X.: Bipartite graph network with adaptive message passing for unbiased scene graph generation. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Los Alamitos, CA, USA, pp 11104–11114. IEEE Computer Society (2021)
14. Li, C., Jia, K., Shen, D., Richard Shi, C.J., Yang, H.: Hierarchical representation learning for bipartite graphs. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, pp 2873–2879. International Joint Conferences on Artificial Intelligence Organization (2019)
15. Ahmed, H., Ali, S.: Research on bipartite network embedding with auxiliary information. In: 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, pp. 1–6. IEEE (2019)
16. Gao, M., He, X., Chen, L., Liu, T., Zhang, J., Zhou, A.: Learning vertex representations for bipartite networks. IEEE Trans. Knowl. Data Eng. **34**(1), 379–393 (2022)
17. Kaya, B.: Hotel recommendation system by bipartite networks and link prediction. J. Inf. Sci. **46**, 016555151882457 (2019)
18. Sheikh, N., Kefato, Z., Montresor, A.: gat2vec: representation learning for attributed graphs. Computing **101**(3), 187–209 (2019)
19. Shen, Y., Traganitis, P.A., Giannakis, G.B.: Nonlinear dimensionality reduction on graphs. CoRR, abs/1801.09390 (2018)
20. Chen, H., Perozzi, B., Al-Rfou, R., Skiena, S.: A tutorial on network embeddings. CoRR, abs/1808.02590 (2018)
21. Cao, S., Lu, W., Xu, Q.: Grarep: Learning graph representations with global structural information. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM '15, New York, NY, USA, pp 891–900. Association for Computing Machinery (2015)
22. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, New York, NY, USA, pp 1105–1114. Association for Computing Machinery (2016)
23. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, Red Hook, NY, USA, pp 3111–3119. Curran Associates Inc (2013)
24. Dong, Y., Chawla, N.V., Swami, A.: Metapath2vec: scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery

and Data Mining, KDD '17, New York, NY, USA, pp. 135–144. Association for Computing Machinery (2017)
25. Fu, T-Y., Lee, W-C., Lei, Z.: Hin2vec: explore meta-paths in heterogeneous information networks for representation learning. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, New York, NY, USA, pp 1797–1806. Association for Computing Machinery (2017)
26. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2017)
27. Kipf, T.N., Welling, M.: Variational graph auto-encoders. Preprint at https://arxiv.org/abs/1611.07308 (2016)
28. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Red Hook, NY, USA, pp. 1025–1035. Curran Associates Inc (2017)
29. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. CoRR, abs/1806.01973 (2018)
30. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, New York, NY, USA, pp. 1225–1234. Association for Computing Machinery (2016)
31. Jia, Y., Lin, M., Wang, Y., Li, J., Chen, K., Siebert, J., Zhang, G.Z., Liao, Q.: Extrapolation over temporal knowledge graph via hyperbolic embedding. CAAI Transactions on Intelligence Technology **n/a**(n/a)
32. Song, X., Li, J., Cai, T., Yang, S., Yang, T., Liu, C.: A survey on deep learning based knowledge tracing. Knowledge-Based Systems **258**:110036
33. Chang, S., Han, W., Tang, J., Qi, G.-J., Aggarwal, C.C., Huang, T.S.: Heterogeneous network embedding via deep architectures. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, New York, NY, USA, pp. 119–128. Association for Computing Machinery (2015)
34. Lu, Y., Shi, C., Hu, L., Liu, Z.: Relation structure-aware heterogeneous information network embedding. CoRR, abs/1905.08027 (2019)
35. Chen, X., Yu, G., Wang, J., Domeniconi, C., Li, Z., Zhang, X.: ActiveHNE: active heterogeneous network embedding. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, pp 2123–2129. International Joint Conferences on Artificial Intelligence Organization (2019)
36. Xu, L., Wei, X., Cao, J., Yu, P.S.: Embedding of embedding (EOE): Joint embedding for coupled heterogeneous networks. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM '17, New York, NY, USA, pp 741–749. Association for Computing Machinery (2017)
37. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. Preprint at http://arxiv.org/abs/1301.3781 (2013)
38. Yoon, Y., Hong, J., Kim, W.: Item recommendation by predicting bipartite network embedding of user preference. Expert Syst. Appl. **151**, 113339 (2020)
39. Cai, X., Zhao, W., Zhao, J., Guan, Z., Song, X., Li, J.: Uncertainty-aware multiview deep learning for internet of things applications. IEEE Trans. Industr. Inf. **19**(2), 1456–1466 (2023)
40. Xu, C., Guan, Z., Zhao, W., Wu, H., Niu, Y., Ling, B.: Adversarial incomplete multi-view clustering. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, pp. 3933–3939. International Joint Conferences on Artificial Intelligence Organization (2019)
41. Xie, Y., Gong, M., Gao, Y., Qin, A.K., Fan, X.: A multi-task representation learning architecture for enhanced graph classification. Front. Neurosci. **13** (2020)
42. Hancock, J.T., Khoshgoftaar, T.M.: Survey on categorical data for neural networks. J. Big Data **7**, 1–41 (2020)
43. Chen, W., Liu, T.-Y., Lan, Y., Ma, Z., Li, H.: Ranking measures and loss functions in learning to rank. In: Proceedings of the 22nd International Conference on Neural Information Processing Systems, NIPS'09, pp. 315–323. Red Hook, NY, USA. Curran Associates Inc (2009)
44. Hou, C., He, S., Tang, K.: Attributed network embedding for incomplete structure information. CoRR, abs/1811.11728 (2018)

Authors and Affiliations

Sajjad Athar[1] · Rabeeh Ayaz Abbasi[1] · Zafar Saeed[2] · Anwar Said[3] · Imran Razzak[4] · Flora D. Salim[4]

Sajjad Athar
sajjad.athar@cs.qau.edu.pk

Zafar Saeed
zafar.saeed@uniba.it

Anwar Said
anwar.said@itu.edu.pk

Imran Razzak
imran.razzak@unsw.edu.au

Flora D. Salim
flora.salimk@unsw.edu.au

1 Quaid-i-Azam University, Islambad, Pakistan

2 Dipartimento di Informatica, Università degli Studi di Bari, Bari, Italy

3 Information Technology University, Lahore, Pakistan

4 University of New South Wales, Sydney, Australia