Social Network Analysis and Mining (2023) 13:54
https://doi.org/10.1007/s13278-023-01058-z

REVIEW PAPER

A survey on bipartite graphs embedding

Edward Giamphy[1,2] · Jean-Loup Guillaume[2] · Antoine Doucet[2] · Kevin Sanchis[1]

Received: 24 November 2022 / Revised: 2 March 2023 / Accepted: 4 March 2023 / Published online: 21 March 2023

Abstract
Research on graph representation learning (a.k.a. embedding) has received great attention in recent years and shows effective results for various types of networks. Nevertheless, few initiatives have been focused on the particular case of embeddings for bipartite graphs. In this paper, we first define the graph embedding problem in the case of bipartite graphs. Next, we propose a taxonomy of approaches used to tackle this problem and draw a description of state-of-the-art methods. Then, we establish their pros and cons with respect to conventional network embeddings. Finally, we provide a description of available resources to lead experiments on the subject.

Keywords Graph embeddings · Bipartite graph · Representation learning · Graph-based pattern representations · Machine learning · Data mining · Survey · Benchmark

1 Context and overview

Networks come as a natural way to model a diverse set of real-world information. They are used in topics as diverse as computational biology, recommendation systems, finance or social networks (Hegeman and Iosup 2018). Networks offer a rich structure that can model complex relationships. However, this advantage may turn as a downside because of the processing capacities that they require. In the era of big data, information networks can contain billions of nodes and edges. Therefore, it can be intractable to perform complex inference procedures on the entire network.

A common way to bypass this limitation is network representation learning, also known as network embedding (Arsov and Mirceva 2019). The main principle here is to reduce the dimension of the network in an embedding space while preserving major network characteristics (structure, nodes' proximity, edges). Precisely, the purpose of embeddings is to find a mapping function which associates a low-dimensional latent representation to each node in the network. Such representations require less effort to be handled; hence, they can be used as features for common tasks on graphs and are directly usable by various downstream machine learning algorithms.

With quantities of data exploding, this area of research has attracted a strong interest from the scientific community and inspired numerous works. In particular, recent years have seen the emergence of several sub-domains of graph embeddings with a particular focus on the considered network type. Consequently, a variety of tailor-made methods have been created to investigate specific network types, e.g., homogeneous networks (Cai et al. 2017), attributed networks (Li et al. 2021), heterogeneous networks (Yang et al. 2020), hyper networks (Ha et al. 2016), dynamic networks (Barros et al. 2021), or bipartite networks (Stauffer et al. 2017).

Bipartite graphs have a particular importance since many real-world applications such as topic modeling, medical diagnosis or recommendation can be modeled as computing on bipartite graphs (see Table 2 for more examples of application). Nevertheless, representation learning for bipartite graphs has only received little attention, with a few papers dedicated to it. In this survey, we try to bring an overview of works related to bipartite graph embeddings and gather

✉ Edward Giamphy
edward.giamphy@preligens.com

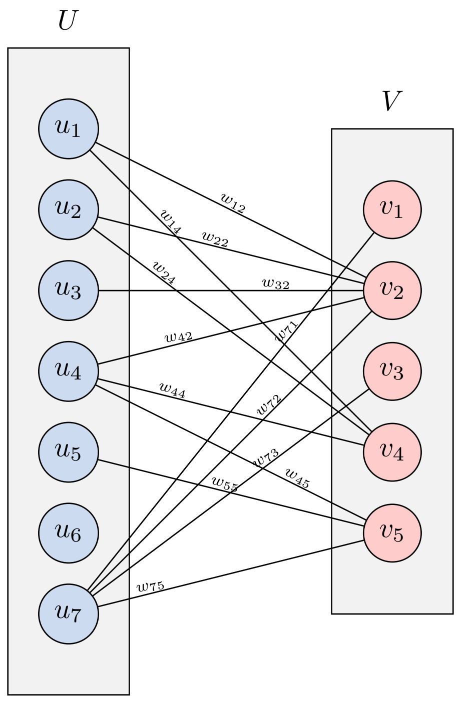
Jean-Loup Guillaume
jean-loup.guillaume@univ-lr.fr

Antoine Doucet
antoine.doucet@univ-lr.fr

Kevin Sanchis
kevin.sanchis@preligens.com

1 AI Research, Preligens, rue de Provence, 75009 Paris, Ile de France, France

2 L3i, La Rochelle University, Avenue Albert Einstein, 17000 La Rochelle, Charente Maritime, France



$$W = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

$$W^T = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Fig. 1 Example of a bipartite graph with the adjacency matrix of both domains U and V on the right side. The positions in the adjacency matrix represent the weights in the bipartite graph. The weight $w_{ij} = 1$ if nodes u_i and v_j are connected, 0 otherwise. In some bipartite graphs, the weights may be any non-negative scalar, hence describing the strength of the relationship between nodes

a list of tools available to explore this area. To the best of our knowledge, this work is the first survey focused on this subject.

The rest of this paper is structured as follows. Section 2 introduces the notations and provides some mathematical background to define the bipartite graph embedding problem. Section 3 describes state-of-the-art methods used to tackle bipartite graph embeddings and categorizes them into a taxonomy of approaches. Moreover, it establishes the pros and cons of each method and points toward directions of future improvements. Section 4 draws a description of available resources to lead experiments on bipartite graph embeddings. Finally, Sect. 5 concludes the paper with a discussion on several open problems and possible research directions.

2 Notations and problem formulation

We hereby introduce notations and definitions of concepts required to formalize the bipartite network embedding problem.

2.1 Notations

Throughout this paper, we use the following notations: scalars are denoted as lowercase italic letters (e.g., s); matrix is denoted as uppercase bold letters (e.g., $\boldsymbol{A}$); the transpose of a matrix $\boldsymbol{A}$ is denoted $\boldsymbol{A^T}$; the element of ith row and jth column is denoted a_{ij}.

Additionally, both 'graph' and 'network' terms will be used to describe the same data structure, i.e., a set of nodes connected by edges.

2.2 Definitions

In this section, we define important terms and definitions used to describe bipartite graph embedding.

- **Homogeneous graph** In this work, we use the term homogeneous graph to refer to simple graph, as opposed to bipartite graph. The term 'homogeneous' is used to highlight that every node of the graph belongs to a same unique type of node. Formally, a simple or homogeneous graph $G = (U, E)$ is defined by a set of nodes U and a set of edges E between these nodes. An edge going from node $u \in U$ to node $v \in U$ is denoted as $(u, v) \in E$. This definition closely follows [9].
- **Bipartite graph** A bipartite graph $G = (U, V, E)$ is composed of two sets of different domains (or types) of nodes U and V, and a set of edges $E \subset U \times V$. Bipartite graphs have the particularity that they only consider inter-domain edges: within a domain (U or V) two nodes cannot be connected. Figure 1 gives an example of bipartite graphs where we can observe the two distinct domains and their connections. Let $n = |U|$ and $m = |V|$. Let u_i be the ith node of U, $i = 1, 2, ..., n$ and v_j be the jth node of V, $j = 1, 2, ..., m$. Each (u_i, v_j) edge may carry a weight w_{ij} describing the strength of the relationship between nodes u_i and v_j. If a graph is unweighted, we can consider that

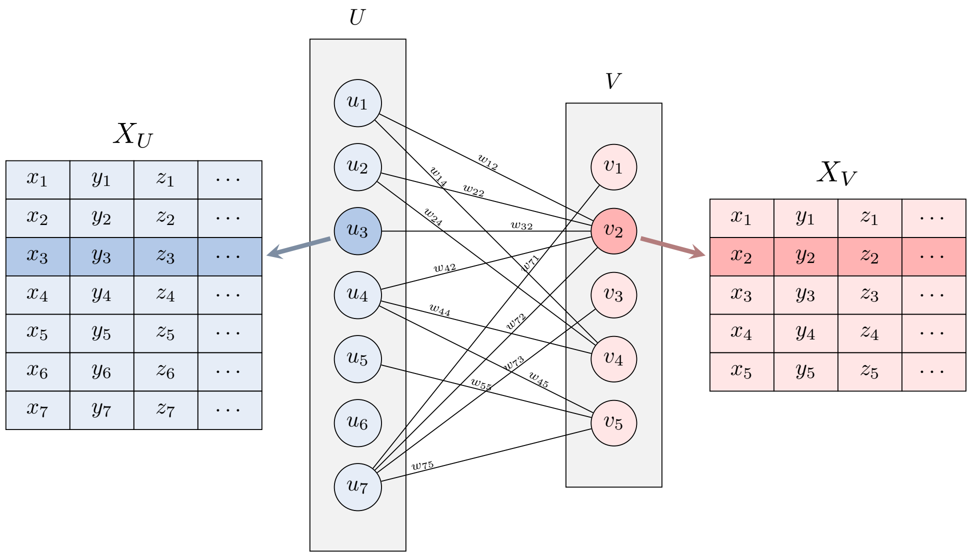

U
V
X_U
X_V
u_1 u_2 u_3 u_4 u_5 u_6 u_7
v_1 v_2 v_3 v_4 v_5
w_{12} w_{14} w_{22} w_{24} w_{32} w_{42} w_{71} w_{44} w_{72} w_{73} w_{45} w_{55} w_{75}
x_1 y_1 z_1 $\cdots$
x_2 y_2 z_2 $\cdots$
x_3 y_3 z_3 $\cdots$
x_4 y_4 z_4 $\cdots$
x_5 y_5 z_5 $\cdots$
x_6 y_6 z_6 $\cdots$
x_7 y_7 z_7 $\cdots$

Fig. 2 Illustration of node features of both domains U and V

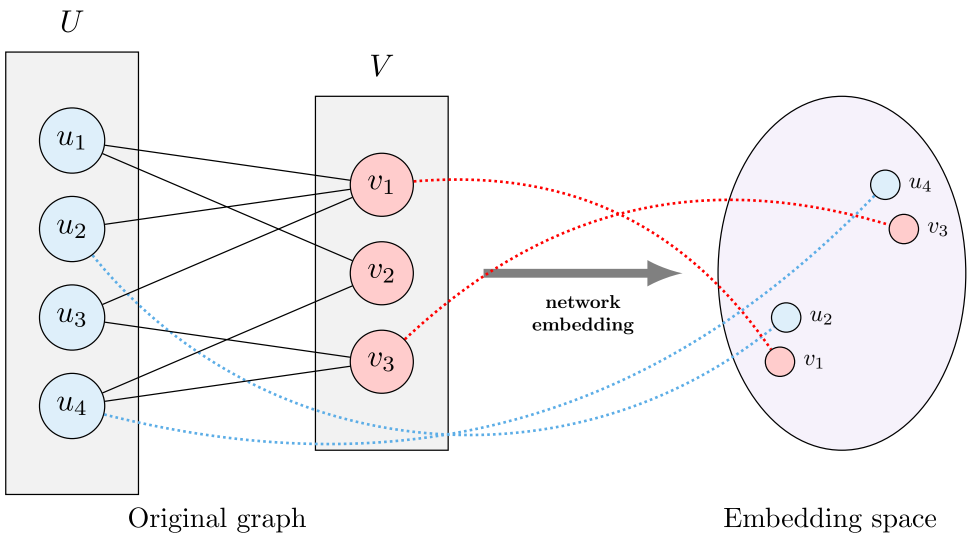


Fig. 3 Illustration of the embedding process for bipartite networks. The embedding operation aims to map nodes to a low dimensional embedding space. One of the challenges is that distances in the embedding space should reflect the node positions in the original graph. For instance, in the original graph, nodes u_2 and v_1 are connected; therefore, their representations should be close in the embedding space. The same goes for node v_3 and u_4

all weights are equal to 1. Likewise, if two nodes *i* and *j* are not connected, we set $w_{ij} = 0$. Therefore, we can use a $n \times m$ matrix $\boldsymbol{W} = (w_{ij})$ to represent all weights in the bipartite network. Precisely, $\boldsymbol{W}$ is named the adjacency matrix of domain U and the adjacency matrix of domain V is the transpose of $\boldsymbol{W}$: $\boldsymbol{W^T} \in \mathbb{R}^{m \times n}$. Graphs may also be associated with attribute or feature information such as profile information associated with an user in an online platform, words contained in an article or movie genres. Table 3 provides details about bipartite graph datasets with examples of real-world nodes' types. Feature information is node-level attributes that are represented with a real-valued matrix $\boldsymbol{X_U} \in \mathbb{R}^{n \times p}$ and $\boldsymbol{X_V} \in \mathbb{R}^{m \times p'}$, where we assume that the ordering of the nodes is consistent with the ordering in the adjacency matrix. For bipartite graphs, each type of node has its own distinct set of attributes. Figure 2 provides an illustration of node features of the domains in the bipartite graph of Fig. 1.

- **Network embedding** The goal of bipartite network embedding (as well as network embedding in general) is to map all nodes of G to a vector in the low-dimensional embedding space $\boldsymbol{H} \in \mathbb{R}^d$, where each node *u* is represented as a *d*-dimensional embedding vector *h*. In general, *d* is considerably smaller than the number of nodes. Figure 3 gives an illustration of the embedding process, where nodes of the bipartite graph are mapped to a low dimensional embedding space. The final purpose of the embedding process is to optimize this mapping so that geometric relationships in the embedding space reflect the structure of the original graph. After optimizing the embedding space, the learned embeddings can be used as feature inputs for downstream machine learning

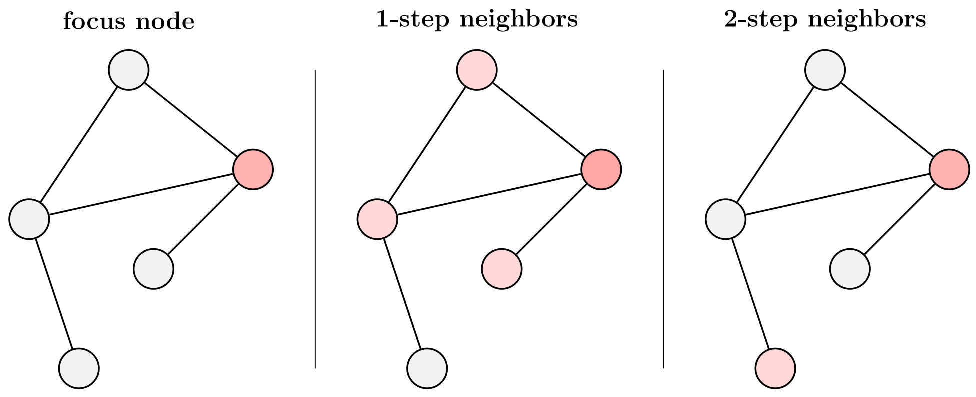


Fig. 4 Illustration of multi-order relationship within a standard graph. The image in the left highlights the node we focus on. The image in the center shows its direct neighbors or 1-step neighbors, while the one on the right shows its first-order neighbor or 2-step neighbor

tasks, such as classification or clustering. This definition closely follows Hamilton et al. (2017).

- **Direct and high order relationship** Direct relationship is defined as any edge from the original bipartite graph, while first-order relationship represents any couple of nodes sharing a common neighbor. In the particular case of bipartite graphs, one can note that nodes in a first-order relationship share the same type. Unlike the previous concepts, two nodes are said to have a second-order relationship if they share a neighbor that shares a neighbor. An illustration of the concept is available in Fig. 4. The image in the center shows direct neighbors or 1-step neighbors, while the one on the right shows a first-order neighbor or 2-step neighbor. One can extend recursively to define higher order relationship. The high-order proximity represents the similarity between the neighbor sets of two nodes. These different relationships are among the characteristics that graph embedding aims to preserve.

3 Taxonomy

In this section, we present and group existing algorithms for bipartite graph embedding into two categories based on shared common characteristics. The categories are inspired by Yang et al. (2020).

The overall purpose of graph embedding methods is to reduce the dimension of high-dimensional graph data into a lower dimensional representation while preserving the desired characteristics of the original graph, such as direct or high order relationship or the heavy tailed distribution of degrees. Since most techniques do not consider the degree distribution, high degree nodes in the network space do not necessarily have a dense neighborhood in the embedding space (and the same applies to low degree nodes). This specific topic has been studied in more details in Liu et al. (2021).

Bipartite graph embedding models also aim to preserve characteristics of the original graph, and it often assumes a huge number of nodes as described by network's number of nodes in Table 3. Thus, further consideration is given to scalability issues.

Due to this assumption, most methods of representation learning for bipartite graphs are unsupervised, or at best self-supervised. Indeed, labels needed for a supervised approach on a network with millions of nodes would require massive efforts of annotation. Furthermore, current supervised methods cannot embed the rich information contained in distinct node features from two domains and topology information into a single node representation (Dwivedi et al. 2022; Gao et al. 2018; Bobadilla et al. 2013).

We provide the following taxonomy for existing bipartite graph embedding algorithms:

1. **Proximity-Preserving Approaches**;
2. **Message-Passing Approaches**.

3.1 Proximity-preserving approaches

This group of methods leverages the local and global structures of a graph in order to preserve relevant characteristics in the embeddings. Figures 5 and 6 illustrate a few problems related to such methods. In order to understand the essence of these methods, consideration must be taken to the notions of direct, first-order and high order relationship that we defined in Sect. 2.2.

Following classical network embedding solutions, a few works have tried to transfer the random walk idea. These methods typically apply a two-step solution. First, random walks are performed from several nodes in order to explore the network. The goal being to establish a "corpus" of nodes. The random walks applied in the network build a list of sequences of nodes, similar to word sentence in Natural Language Processing (NLP) domain, that forms the corpus of visited nodes. Then, in the second step, a word embedding method (such as word2vec (Mikolov et al. 2013)) is applied

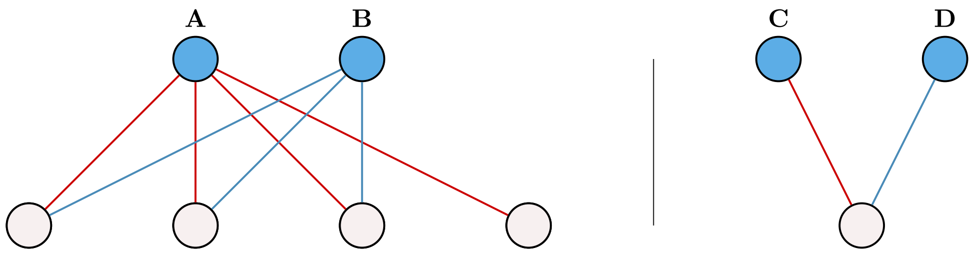


Fig. 5 Example of structural consideration within networks. The figure represents different patterns of connection between nodes. On the left side of the figure, nodes A and B have 3 common neighbors, whereas in the right part of the figure, nodes C and D share one common neighbor. Different methods are available to evaluate which of the pairs A-B or C-D have the strongest relationship. One could argue that nodes A and B share more neighbors than nodes C and D and therefore, should be considered as the strongest connection. But on the other hand, one could argue that node C and D share exactly the same neighbors unlike A and B and should be considered as a stronger connection

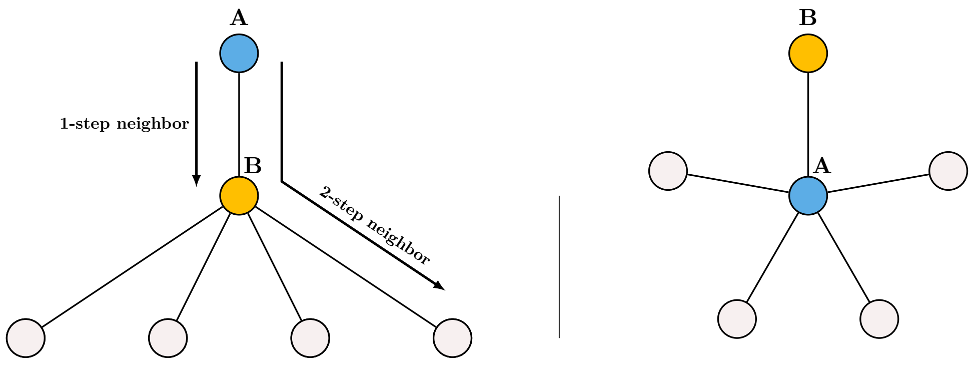


Fig. 6 The left side figure represents a graph where node A owns one 1-step and four 2-step neighbors, while the right side figure represents a graph where node A owns five 1-step neighbors. If 1-step and 2-step neighbors are treated equally, i.e., if distance is not considered, then these two structurally different graphs are perceived similarly by node A. This is counter-intuitive and shows how k-step neighbors processing can impact graph structural considerations

to the nodes sequences to obtain the embeddings of nodes. These methods can be suboptimal for embedding bipartite networks due to two reasons:

1. Most of the time the two nodes' domains of the bipartite graph are treated equally, and no distinction is done between the node information of the two domains;
2. The generated corpus may not preserve the characteristics of a bipartite network, such as the long tail distribution of nodes' degree (Guillaume and Latapy 2004) or the power-law distribution of real-world bipartite graphs (Chakrabarti and Faloutsos 2006).

BiNE (Gao et al. 2018) is a random walk-based method. It adheres to a group of methods inspired by the pioneer works of DeepWalk (Perozzi 2014) and node2vec (Grover and Leskovec 2016) that adapt the idea of Skip-gram (Mikolov et al. 2013) to model homogeneous networks, and try to extend this method to bipartite networks.

BiNE performs biased random walks purposefully in order to preserve some properties of the bipartite graph. One of the key objectives of BiNE is to maintain the degree distribution of a network (Guillaume and Latapy 2004; Newman et al. 2001). To do so, BiNE takes careful consideration in modeling explicit relations as well as implicit relations. Explicit relations aim to reconstruct the bipartite network by targeting the observed links (direct links between nodes), while the implicit relations capture the high-order correlations (second-order relationship between nodes or unobserved links). To conserve the main characteristics of the bipartite network, BiNE performs biased random walks that adapt themselves to the node where they start, based on the importance of the latter. Specifically, the number of random walks and their length are defined depending on said importance. This way the node distribution in the generated corpus is more consistent with the original bipartite network.

BiNE stands out of regular network embedding methods because of its two contributions. The first one is the incorporation of implicit relations which have shown improvement of embeddings (Yu et al. 2018; Jiang et al. 2016). The second one is the definition of a joint optimization framework to account for both explicit and implicit relations. A dedicated objective function is defined for every relation, and the node embeddings of each objective function are gathered to reinforce each other.

BiANE (Huang et al. 2020) aims to improve the consideration of the relationship between the attribute information and the structure information of nodes, often neglected. Indeed, both information represent different but complementary aspects of the nodes; however, most works about network embeddings overlook their different properties. To do so, the main contribution of BiANE is the introduction of an innovative way to take into account high-order proximity structure without adding scalability difficulties to the calculation process. BiANE defines the inter-domain proximity as the links that exist across the two domains of the bipartite graph and the intra-domain proximity as the underlying proximity within each domain (e.g., the proximity between nodes of the same type).

In details, BiANE extracts and isolates intra-domain and inter-domain proximities. It then uses an autoencoder (Rumelhart and Mcclelland 1986) to get an embedding in a latent space. Specifically, given a bipartite attributed network, BiANE first partitions the network according to nodes' type or domain. Then, it extracts the structure information and the attribute information from each partition and compresses them through different auto-encoders. Next, it integrates the first-order proximity within each partition. At that point, BiANE performs dynamic positive sampling in the latent space and proposes a novel correlation training to enhance the correlation between attribute information and structure information. Modeling the high-order structure proximity may be a tricky problem as existing works (Gao et al. 2018, 2019) add large amounts of additional links to do so, but they make the network much denser and introduce scalability problems. Finally, after performing dynamic positive sampling (Malkov and Yashunin 2016), BiANE aggregates the encoding vectors from the aforementioned autoencoders and models the inter-domain proximity of the bipartite attributed network. BiANE offers a convenient way to preserve both intra-domain and inter-domain proximities while introducing an efficient dynamic positive sampling strategy that improves the training process. Nevertheless, further experiments should be conducted to validate the dynamic positive sampling strategy used by the authors. In fact, the positive sampling strategy is different from traditional sampling method.

FOBE and HOBE [28] aim to preserve the characteristics of the close and distant neighborhoods of bipartite graphs. It introduces two bipartite embedding models that learn dense latent representations of bipartite graphs while preserving type-specific semantic information. Both methods share the same approach. They first observe structural relationship by sampling on certain types of relationship between nodes. Then, they learn an embedding to minimize the difference between the observed nodes' similarities and their corresponding estimated similarities. Each of the methods differs in the way they lead theirs observations, estimations and objectives. The major contribution of FOBE and HOBE is that they generate estimations of similarities within each node domain separately and thus, allow to better preserve type-specific latent features.

Technically, FOBE first samples direct links and first-order relationships between nodes. Then, it learns embeddings by minimizing the Kullback–Leibler Divergence (Shlens 2014) between observations and embeddings estimations. On the other hand, HOBE first computes algebraic similarities for pairs of nodes of all edges. The algebraic distance (Chen and Safro 2011) has proven to be a good mean to capture implicit similarities in hypergraphs. The computation of algebraic distance allows HOBE to take into account second-order relationships in addition to direct and first-order relationships. Then, HOBE fits the embeddings with mean-squared error.

FOBE retains local relationships, while HOBE captures higher-order relationships. Therefore, in order to unify these approaches, a method of combination is proposed to join the different embeddings into a 'best of both world' embeddings. Two versions of the combination method are suggested. The first version fosters the maximization of performance on the training task (the direct version), while the second exploits the full encoding of input embeddings (the auto-regularized version).

3.2 Message-passing approaches

Message-Passing-based methods rely on a simple intuition: for every node, an embedding is iteratively built based on the node's feature and the aggregated information of its local neighborhood.

Precisely, two differentiable functions, an 'update' function and an 'aggregate' function, are defined and will be used at each iteration of the embedding process. For each node, at each iteration, the aggregate function takes as input the set of embeddings of the node's neighborhood and outputs a message gathering information about this neighborhood. Then, the update function merges this message with the previous embedding of the node to produce the updated embedding, and the process is then repeated a finite number of times. Figure 7 illustrates this process. The Message-Passing mechanism is at the root of the Graph Neural Networks (GNNs) framework for defining deep neural networks on graph data [9] and inspires numerous works of graph representation (Wu et al. 2021).

BGCN [32] extends the notion of Graph Convolutional Networks (GCNs) (Kipf and Welling 2016) to bipartite networks. Although GCNs are usually limited to homogeneous network, BGCN extends it to bipartite networks by adapting

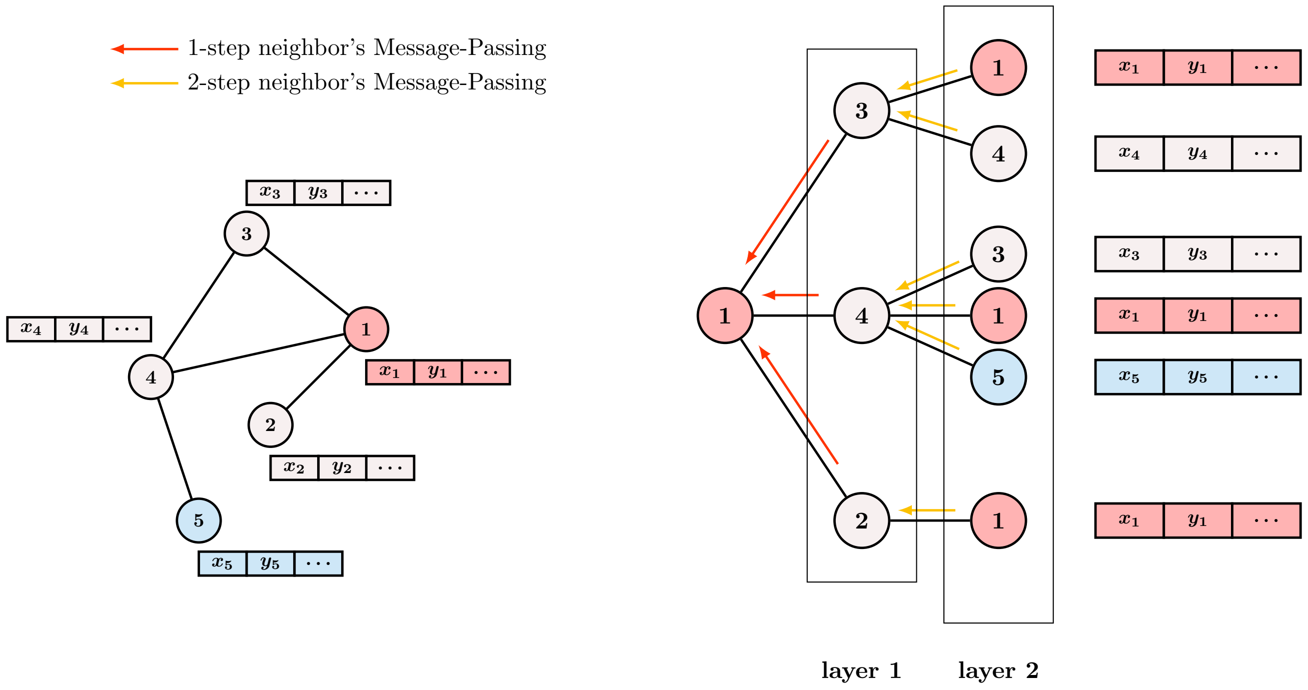


Fig. 7 Illustration of the message passing process. The example shows a 2 steps depth message passing process for node 1. Starting from node 1, the layer 1 contains all the direct neighbors of node 1, namely nodes 3, 4 and 2. Then, the same operation is repeated for every node in the layer 1. For example, all direct neighbors of node 3 (in layer 1) are put into layer 2; hence, one can see nodes 1 and 4 in the upper part of layer 2. To get the other nodes of layer 2, the same principle is repeated for the other nodes of layer 1. The arrows converging toward a node embed its update and the transmission of its neighbors' information. This takes place into two distinct operations. First, all neighbors' features are aggregated by an aggregation function. Then, the node's state is updated by an update function that takes into account both the features' aggregation and the previous state of the node

the equations of network representation learning to the specificity of bipartite network matrix notations. Then, it applies an embedding function composed of 2 elements: a one-hop neighbor features aggregation and a nonlinear transformation that take as input the aforementioned aggregation. Next, after defining the concept of GCN for bipartite networks, the paper examines two types of decoder architectures. The first decoder is a standard multi-layer perceptron (MLP) model that aligns the output from the embedding function with each node feature through two fully connected neural network layers. The second decoder is based on an adversarial approach: it trains a discriminator to separate between elements from the original feature matrix and the embedding vectors. The discriminator aims to maximize the ability to identify the feature representation, and BGCN aims to prevent the discriminator from doing so by generating encoded representations that are as similar as possible.

Finally, in order to go beyond the one-hop neighbor information limit of the previous models, a cascaded architecture based on the BGCN-decoder is introduced to capture the multi-hops structure information.

Bi-HGNN [32] proposes a recommender system for an e-commerce platform that utilizes both the users' community-level generalization as well as their individualized preferences. The model is part of hierarchical message-passing models but sets itself apart by considering the user-specific information, often ignored in regular models, that is not captured by the community-level generalization.

The proposed method tackles the drawback of community-level generalization encountered in recommender systems where individualized preferences are not reflected in the communities the users belong to. To do so, it draws inspiration from works on hierarchical neural networks, commonly used for the task of graph classification, and apply such concepts on the task of link prediction.

Bi-HGNN first uses a standard GNN model (Hamilton et al. 2018) to model users and items in a bipartite graph and encodes users and items' raw features using Wide and Deep (Cheng et al. 2016). Bi-HGNN relies on predefined communities of nodes of users and assigns low-dimensional embeddings to these communities. Then, it uses said embeddings to perform a clustering on every user node. In details, the model computes the distance between nodes of users and every community by using the aforementioned embeddings. This distance metric allows to softly assign each node of users into a few communities. Then, the user node information is decomposed into two orthogonal spaces representing the information captured by community-level generalization and individualized users' preferences.

Cascade-BGNN (He et al. 2020) is a self-supervised representation learning framework for large-scale bipartite graphs. There are three key designs within the architecture of the embedding model. The first is inter-domain message passing (IDMP). Its goal is for one domain to aggregate information from the other domain through the connected edges. IDMP only performs aggregation on each node's neighbor nodes without involving the node itself.

Once the aggregated features from the opposite domain are attached, intra-domain alignment (IDA) is used to fuse these two distinct features into a single representation. As IDMP does not include the information of the node itself, IDA allows to take into account the node features. After one layer is trained through minimization of an adversarial loss for both domains in a self-supervised manner, the representation of the two domains is obtained.

The embedding thus obtained merely captures the one-hop topology structure between domains as well as feature information from both domains. However, one-hop aggregation does not sufficiently characterize diverse graph structures; hence, a multi-hop mechanism is required. Instead of leveraging the typical end-to-end training method, this paper develops a cascaded training method to drive multi-hop message passing. In details, cascaded training is the concatenation of several basic BGNN blocks, and a basic BGNN block is the sequence of IDMP with IDA. A basic BGNN block constitutes a one depth training and completes its training with one-hop embedding. Then, the trained embedding is used as the input for later training (e.g., as input for the next BGNN block).

HGCN (He 2019) proposes an extension of convolutional model to learn graph representations of bipartite networks. Regular convolution-based models are limited to homogeneous graph, and HGCN is an attempt to overcome this boundary. Moreover, many random walk algorithms may not preserve the power-law distribution of real-world bipartite graphs (see Sect. 3.1), which causes a loss of information in high degree nodes. HGCN has been designed to address a few challenges related to bipartite graphs. First, it is adapted to large-scale graphs and to deal with heterogeneous edge and node features. Plus, it is suited for unsupervised learning and takes into account nodes' types in bipartite graphs.

This approach proposes a three-step cascaded HGCN architecture in order to capture both explicit and implicit relations between the two domains of the bipartite graph. The first two steps are symmetric and include, respectively, feature information from one domain and structural information from the opposite domain. The third step then combines their output. Specifically, the first step consists in capturing the explicit representation by learning node representations from the first domain while including neighbor features of the second domain. The second step captures the implicit relation by learning node representations from the second domain while including neighbor features of the first domain. Finally, the third step merges both the implicit and explicit relations. Two output models are then proposed: a decoder model and an adversarial learning model. A drawback of HGCN is the lack of consistency of the method used to fuse nodes feature information with structural information.

IGE (Zhang et al. 2017) generalizes embedding techniques to the challenging case of dynamic bipartite graphs with attributed edges, or attributed interaction graphs as called by the paper. In attributed interaction graphs, an edge represents an interaction between two nodes of different domain, and attributes of the edge represent the content of the interaction. The stock market may be taken as an example for a better understanding. Each transaction involves an investor that buys or sells a stock at a certain price, in a certain quantity and at a certain time. Here, the nodes of the bipartite graph are the investors and the stocks, the edges between them are the transactions and the transactions' characteristics are the attributes of the edge.

IGE bypasses the difficulties of dynamics and heterogeneous attributes of edges by investigating the temporal dependency of edges instead of the structure of graphs. To do so, it introduces a deep node embedding method. IGE consists of a three neural networks architecture containing two coupled multiplicative neural networks for prediction and an attributes' encoding network. The encoding network transform attributes into homogeneous fixed-length vectors in order to overcome the heterogeneity of attributes. Then, the homogeneous encoded attributes are provided to the coupled prediction networks that investigate the temporal dependency by learning node embeddings for interaction graphs.

3.3 Discussion

The aforementioned methods have limits that we summarize and discuss in this section.

FOBE and HOBE bring an innovative approach, but few consideration is given to nodes features in the embedding. Plus, the scalability of the algorithms is not discussed.

Despite its advantages and good results on both link prediction and recommendation tasks, BiNE suffers from some limits. Indeed, it relies exclusively on observed edges and thus, may encounter difficulty in reconstituting nodes that have very few or even no connections. This obstacle may be addressed by including auxiliary side information, such as numerical features or textual descriptions, into the model. Finally, a major axis of improvement for BiNE is to take into account the dynamic aspect of networks.

BGCN and Cascade-BGNN are relevant methods, but the advantages of the cascaded architecture over traditional methods are not explicit, except the efficiency.

Table 1 Evaluation tasks used by bipartite graphs embedding models

Method	Category	Node classification	Relation prediction
BiNE	Random walk		✓
BGCN	GNN	✓	
Bi-HGNN	GNN		✓
BiANE	Random walk	✓	✓
FOBE/HOBE	Proximity based		✓
Cascade-BGNN	GNN	✓	
IGE	GNN	✓	
HGCN	GNN	✓	

Bi-HGNN proposes an interesting way to overcome a well-known limitation in regular recommender systems (community-level generalization), but the adopted point of view is too geared toward e-commerce platforms. Thus, the brought innovation is not obvious and the links with general embeddings issues are sometimes hidden. Moreover, the scalability potential of Bi-HGNN lacks details.

Similarly to other methods of this work, IGE focuses on node embeddings without incorporating edge embeddings. However, in the case of IGE, which considers attributed interaction graphs, this choice is questionable as the edges of such graphs carry more information than regular bipartite graphs. This constitutes an improvement axis of the IGE method. Besides, IGE only uses nodes information and does not leverage structural information, which may be another improvement axis. Finally, a general drawback of IGE is that it does not embed nodes of different type in the same latent space. Thus, a major enhancement would be to remove this drawback by mapping nodes of different types in the same space.

The limitations pointed above show that bipartite graph embedding suffers from scalability issues and from coupling heterogeneous information which should motivate future works in the area. Moreover, these limitations do not decide between the categories, but the proposed taxonomy shows a clear trend toward Message-Passing approaches in the scientific community.

4 Tools for evaluation and experimentation

In this section, we list a set of commonly used tools to explore the domain of representation learning for bipartite networks and provide a list of online resources to lead works and experiments. This list is composed of libraries as well as datasets that are conventionally used in papers in this area. Table 1 presents the tasks that have been previously identified in the literature.

4.1 Evaluation tasks

In the vast majority of current graph applications, representation learning is used for four tasks: node classification, graph classification, relation prediction or network reconstruction. These tasks allow to cover a large range of real-world applications. For an exhaustive explanation of machine learning tasks on graphs, and how they interact with the standard machine learning categories, the interested reader may refer to Chapter 1.2 of [9].

4.1.1 Node classification

Node classification is a machine learning task on graph data and has gained popularity in recent years. Node classification is about building a model able to classify nodes into pre-identified categories. Given only a small number of manually labeled examples from the entire network, the goal is to predict the label associated with all the nodes of the network. Node classification seems very similar to traditional supervised classification, but a major distinction remains that nodes in a network are not independent and identically distributed. Indeed, node classification breaks this i.i.d. assumption because the goal of network analysis is to model an interconnected, and thus, dependent set of nodes. Examples of node classification application are any kind of community detection. Node classification may be used in social network to classify similar users into the same community (Min et al. 2021), or in a document database to classify document sharing related content.

4.1.2 Graph classification

Graph classification is a common task from Graph Theory as it aims to characterize entire graphs. The idea is to learn over graph data. Instead of predicting over the individual components (nodes and edges) of a unique graph, a graph classification model takes as input a dataset of multiple different graphs and perform independent predictions for each graph. Here, each graph is an i.i.d. datapoint associated with a label and the goal is to use a training set of labeled data to learn a mapping and predict labels. Examples of graph classification application include molecular property modeling. For instance, in enzyme identification, a collection of graphs, each representing a chemical compound, is used to train a model to predict whether a graph is an enzyme or not. The same holds for drug discovery, where molecular graphs are considered to identity desired pharmacological and ADME/T (absorption, distribution, metabolism, excretion, and toxicity) properties (Basile et al. 2019). However, the algorithms described in this work do not treat the graph classification task which constitutes a major axis of exploration for future research on the bipartite graph embedding topic.

4.1.3 Relation prediction

Relation prediction, also known as link prediction or graph completion, is among the most popular machine learning tasks on graph. In the standard setup for relation prediction, given a set of nodes and an incomplete set of edges between these nodes, the goal of relation prediction is to use the partial information available to infer the missing edges of the graph. The complexity of relation prediction is highly correlated with the type of analyzed graph data. For instance, in simple graphs (e.g., "friendship" social network) there are simple heuristics based on how many neighbors two nodes share, whereas in more complex multi-relational graph (e.g., biomedical knowledge graphs with hundreds of biological different interactions), complex reasoning and inference strategies are exploited. Some examples of relation prediction application are recommender systems (Gao et al. 2021). Relation prediction is used for content recommendation in social networks and product recommendation in online shopping platform.

4.1.4 Network reconstruction

As network embeddings may be interpreted as network compression, an accurate compression should be able to reconstruct the network from the embeddings. The network reconstruction task uses the embeddings to predict the original links of the network. This task is closely related to relation prediction task, but network reconstruction uses the existing links of the network as ground truth, whereas relation prediction aims at predicting the probability of plausible link formation. For example, network reconstruction is used in the study of climate changes, to reveal the atmospheric teleconnection patterns and understand their underlying mechanisms, or for identifying functional properties of genes, where gene regulatory networks are predicted from expression data (Gardner et al. 2003). Network reconstruction task is not popular in the literature and is not represented in table 1.

4.2 Benchmark datasets

This section presents several tables providing information on the datasets used by the studied articles.

Table 1 describes the evaluation tasks used in the methods of the taxonomy. It provides information on popularity of evaluation tasks in the recent literature. Table 2 gathers high level information on the bipartite graph datasets. In detail, it gives the kind of graph type, the evaluation tasks the graphs have been used for and the methods in which the graphs have been used. Table 3 summarizes statistics as well as descriptive elements about the nature of the bipartite graph datasets. This is the more furnished table related to the dataset and should be considered as the reference point to have a global view of datasets.

4.3 Open-source libraries

We present here several open-source libraries used for network embedding in general and that can be applied to bipartite network embedding. Furthermore, Table 4 provides links to the source code of the bipartite graphs embeddings models described in Sect. 3.

4.3.1 Pytorch geometric

Pytorch Geometric [69] consists of various methods for deep learning on graphs and other irregular structures, also known as geometric deep learning, from a variety of published papers. In addition, it consists of an easy-to-use mini-batch loader for many small and single giant graphs, multi gpu-support, a large number of common benchmark datasets (based on simple interfaces to create your own), and helpful transforms, both for learning on arbitrary graphs as well as on 3D meshes or point clouds.

4.3.2 DGL—Deep graph library

DGL (Wang et al. 2019) aims to bring graphs closer to deep learning researchers by making easy to implement models from the Graph Neural Networks family. It also intents to make the combination of graph-based modules and tensor-based modules (e.g., PyTorch or MXNet) as smooth as possible.

4.3.3 Spektral

Spektral (Grattarola and Alippi 2020) is a Python library for graph deep learning, based on the Keras API and TensorFlow 2. The main goal of this project is to provide a simple but flexible framework for creating Graph Neural Networks. Spektral may be used to classify users of a social network, predict molecular properties, generate new graphs with Generative Adversarial Networks (GANs), clustering nodes, predict links, and any other task where data are described by graphs.

Table 2 Information about the datasets, including its reference, network type and evaluation tasks. The Evaluation task column follows the legend: A-Node classification, B-Relation prediction. The table also provides which bipartite graphs embedding models have used the datasets. The datasets with no checkmark have been used in (Chen et al. 2019)

Dataset	Network type	Evaluation task	BiNE	BGCN	Bi-HGNN	BiANE	FOBE/HOBE	Cascade-BGNN	IGE	HGCN
Tencent [44]	Online platform	A, B	✓	✓				✓		✓
Cora (McCallum et al. 2004)	Citation network	A		✓				✓		
Citeseer (Giles et al. 1998)	Citation network	A		✓				✓		
MovieLens (Harper and Konstan 2015)	Movie ratings	A, B	✓		✓	✓				
PubMed (Sen et al. 2008)	Citation network	A		✓				✓		
DBLP (Sen et al. 2008)	Citation network	A	✓				✓		✓	
LastFM (Bertin-Mahieux et al. 2011)	Listening records	B					✓			
Amazon (Leskovec et al. 2007)	Product co-purchasing	B					✓			
YouTube [51]	Group membership	B	✓				✓			
Friendster [51]	Group membership	B					✓			
Livejournal graphs (Backstrom et al. 2006; Leskovec et al. 2008)	Group membership	B					✓			
AMiner (Tang et al. 2008)	Publication network	A, B				✓				
Frappe (Baltrunas et al. 2015)	Application clicks	B								
CiteULike (Wang et al. 2013)	References	B								
Netflix (Bennett and Lanning 2007)	Movie ratings	B								
MovieLens-Latest (Harper et al. 2015)	Movie ratings	A, B								
Last.fm-360K (Celma 2010)	Artist plays	B								
Amazon-Book (He and McAuley 2016)	Book ratings	B								
Epinions-Extend (Tang et al. 2012)	Product ratings	B								
Echonest (McFee et al. 2012)	Song plays	B								
PPD (Wang et al. 2017)	Investing records	A							✓	
Stock (Xu et al. 2018)	Stock exchanges	A							✓	
Yelp (Asghar 2016)	Online reviews	A							✓	
Wikipedia (West and Leskovec 2012; West et al. 2009)	N/A	B	✓							

Table 3 This table summarizes statistics as well as descriptive elements about the nature of the bipartite graph datasets.

Dataset	Node type 1	Node type 2	Edge type	#Class node 1	#Class node 2	#Edges/#Nodes	#Node 1	#Node 2	#Edges
Tencent [44]	User	Community	1 (behavior record)	2	N/A	1.399	619,030	90,044	991,734
Cora (McCallum et al. 2004)	Paper	Paper	1 (citation)	7	6	1.119	734	877	1,802
Citeseer (Giles et al. 1998)	Paper	Paper	1 (citation)	6	6	0.890	613	510	1,000
MovieLens (Harper and Konstan 2015)	User	Movie	5 (5-star)	1	1	111.607	162,000	62,000	25,000,000
PubMed (Sen et al. 2008)	Paper	Paper	1 (citation)	3	3	1.114	13,424	3,435	18,782
DBLP (Sen et al. 2008)	Author	Conference	1 (publication)	5	1	2.908	35,851	20	104,325
LastFM (Bertin-Mahieux et al. 2011)	Artist	User	1 (records)	1	1	4.755	17,632	1,892	92,834
Amazon (Leskovec et al. 2007)	Product	Product	1 (co-purchasing)	1	1	4.711	262,111	N/A	1,234,877
YouTube [51]	User	Group	1 (membership)	1	1	2.613	1,134,890	8,385	2,987,624
Friendster [51]	User	Group	1 (membership)	1	1	27.132	65,608,366	957,154	1,806,067,135
Livejournal graphs (Backstrom et al. 2006; Leskovec et al. 2008)	User	Group	1 (membership)	1	1	8.093	3,997,962	287,512	34,681,189
AMiner (Tang et al. 2008)	Paper	Author	1 (authorship)	1	1	1.150	80,461	66,107	168,525
Frappe (Baltrunas et al. 2015)	User	Item	1 (users' entries)	1	1	19.092	957	4,082	96,203
CiteULike (Wang et al. 2013)	User	Item	2 (like/dislike)	1	1	9.343	5,551	16,980	210,504
Netflix (Bennett and Lanning 2007)	User	Movie	5 (5-star)	1	1	200.820	480,189	17,770	100,000,000
MovieLens-Latest (Harper et al. 2015)	User	Movie	5 (5-star)	1	1	79.881	280,000	58,000	27,000,000
Last.fm-360K (Celma 2010)	User	Artist	many (play count)	1	2	26.876	359,347	294,015	17,559,530
Amazon-Book (He and McAuley 2016)	User	Item	1 (feedback)	2	2	1.970	133,960	431,827	1,114,563

Table 3 (continued)

Dataset	Node type 1	Node type 2	Edge type	#Class node 1	#Class node 2	#Edges/#Nodes	#Node 1	#Node 2	#Edges
Epinions-Extend (Tang et al. 2012)	User	Item	1 (rating)	1	27	2.896	22,166	296,277	922,267
Echonest (McFee et al. 2012)	User	Song	1 (play count)	1	1	34.457	1,019,318	384,546	48,373,586
PPD (Wang et al. 2017)	Investor	Loan	1 (investment)	1	6	16.181	9292	4,501	223,190
Stock (Xu et al. 2018)	Investor	Stock	2 (buy/sell)	3	1	2.916	22,001	939	66,890
Yelp (Asghar 2016)	User	Business	1 (review)	1	10	3.329	724,884	15,726	2,465,173
Wikipedia (West and Leskovec 2012; West et al. 2009)	Article	Navigation path	1 (hyperlink click)	1	2	1.488	4604	76,193	119,882

It provides useful information about graph size, density and its applications.

Table 4 Source code of bipartite graphs embedding models

Method	Source code
BiNE	https://github.com/clhchtcjj/BiNE
BGCN	https://tinyurl.com/ABCGraph
Bi-HGNN	No code available
BiANE	No code available
FOBE/HOBE	http://bit.ly/fobe_hobe_code
Cascade-BGNN	https://github.com/chaoyanghe/bipartite-graph-learning
IGE	No code available
HGCN	https://github.com/chaoyanghe/bipartite-graph-learning

4.3.4 Graph nets library

Graph Nets [72] is DeepMind's[1] library for building GNNs in TensorFlow and Sonnet. The library comes with demos which show how to create, manipulate, and train graph networks to reason about graph-structured data, on a shortest path-finding task, a sorting task, and a physical prediction task.

4.3.5 Jraph—A library for graph neural networks in jax

Jraph [73] (pronounced 'giraffe') is a lightweight library for working with graph neural networks in jax.[2] It provides a data structure for graphs, a set of utilities for working with graphs, and a 'zoo' of forkable graph neural network models.

5 Conclusion and future directions

In this work, we conducted a comprehensive survey of the literature on bipartite graph representation learning techniques. We analyzed recent approaches of bipartite graph embedding that transform a bipartite graph into a low-dimensional vector representation while preserving intrinsic properties of the original graph. We suggested a taxonomy of bipartite graph embedding algorithms that gathers the algorithms by common principles. We described the most common evaluation tasks to compare different bipartite graph embedding methods and to draw insights from algorithms performance. We also gathered lists of datasets and existing libraries to ease the implementation of further research on the subject.

In order to inspire future research on the topic, we now briefly discuss specific directions potentially worth pursuing.

Coupling heterogeneous information Bipartite graphs are constituted by nature with disparate components that make them hard to represent. Bipartite graph embedding methods

[1] https://www.deepmind.com/.

[2] https://github.com/google/jax.

must take into consideration both information at the level of nodes (type, edge information, etc.) and at the level of the graph as a whole. Merging these different types of information is not an easy task and effort must be done to preserve this information in the final representation of the bipartite graph.

Scalability of graph embedding Graphs often are large structures that tend to grow at a rapid pace. We expect to see more and more graphs of a larger scale. Therefore, scalability is central and bipartite graph embedding methods must be designed to deal with this aspect. If a compromise between scalability and quality of the embedding cannot be found, then scalability is an essential characteristic that can discriminate between usable and non-usable approaches.

Dynamic graph embedding Existing works on bipartite graph embedding have barely treated the case of dynamic bipartite graphs (i.e., graphs that evolve through time). However, a lot of real-world graphs are dynamic, such as social graphs. Even if bipartite graph embedding is a nascent area, modeling dynamic mechanisms in the embedding methods is an important research topic as it finds industrial applications. An option to do that is to use bipartite graph with attributed edges, such as Zhang et al. (2017), and to consider time information in the edges of the bipartite graph.

References

Arsov N, Mirceva G (2019) Network Embedding: An Overview. arXiv. arxiv:1911.11726
Asghar N (2016) Yelp dataset challenge: review rating prediction. arXiv . arxiv:1605.05362
Backstrom L, Huttenlocher D, Kleinberg J, Lan X (2006) Group formation in large social networks: Membership, growth, and evolution. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '06, pp 44–54. Association for Computing Machinery, New York, NY, USA . https://doi.org/10.1145/1150402.1150412
Baltrunas L, Church K, Karatzoglou A, Oliver N (2015) Frappe: understanding the usage and perception of mobile app recommendations in-the-wild. arXiv . arxiv:1505.03014
Barros CDT, Mendonça MRF, Vieira AB, Ziviani A (2021) A survey on embedding dynamic graphs. arXiv . arxiv:2101.01229
Basile AO, Yahi A, Tatonetti NP (2019) Artificial intelligence for drug toxicity and safety. Trends Pharmacol Sci 40(9):624–635. https://doi.org/10.1016/j.tips.2019.07.005
Bennett J, Lanning S (2007) The netflix prize
Bertin-Mahieux T, Ellis DPW, Whitman B, Lamere P (2011) The million song dataset. In: Proceedings of the 12th international conference on music information retrieval (ISMIR 2011)
Bobadilla J, Ortega F, Hernando A, Gutiérrez A (2013) Recommender systems survey. Knowl-Based Syst 46:109–132. https://doi.org/10.1016/j.knosys.2013.03.012
Cai H, Zheng VW, Chang KC-C (2017) A comprehensive survey of graph embedding: problems, techniques and applications. arXiv . arxiv:1709.07604
Celma O (2010) Music recommendation and discovery in the long tail. Springer, ???
Chakrabarti D, Faloutsos C (2006) Graph mining: laws generators and algorithms. ACM Comput Surv 38(1):2. https://doi.org/10.1145/1132952.1132954
Cheng H-T, Koc L, Harmsen J, Shaked T, Chandra T, Aradhye H, Anderson G, Corrado G, Chai W, Ispir M, Anil R, Haque Z, Hong L, Jain V, Liu X, Shah H (2016) Wide & deep learning for recommender systems
Chen J, Safro I (2011) Algebraic distance on graphs. SIAM J Sci Comput 33(6):3468–3490
Chen C-M, Wang C-J, Tsai M-F, Yang Y-H (2019) Collaborative similarity embedding for recommender systems. arXiv . arxiv:1902.06188
Dwivedi VP, Luu AT, Laurent T, Bengio Y, Bresson X (2022) Graph neural networks with learnable structural and positional representations. In: International conference on learning representations . https://openreview.net/forum?id=wTTjnvGphYj
Gao M, Chen L, He X, Zhou A (2018) Bine: bipartite network embedding. In: The 41st International ACM SIGIR conference on research & development in information retrieval, pp 715–724
Gao M, He X, Chen L, Liu T, Zhang J, Zhou A (2019) Learning vertex representations for bipartite networks. arXiv . arXiv:1901.09676
Gao C, Zheng Y, Li N, Li Y, Qin Y, Piao J, Quan Y, Chang J, Jin D, He X, Li Y (2021) A survey of graph neural networks for recommender systems: challenges, methods, and directions. arXiv . arxiv:2109.12843
Gardner TS, di Bernardo D, Lorenz D, Collins JJ (2003) Inferring genetic networks and identifying compound mode of action via expression profiling. Science 301(5629):102–105. https://doi.org/10.1126/science.1081900
Giles CL, Bollacker KD, Lawrence S (1998) Citeseer: an automatic citation indexing system. In: DL '98
Godwin J, Keck T, Battaglia P, Bapst V, Kipf T, Li Y, Stachenfeld K, Veličković P, Sanchez-Gonzalez A Jraph: a library for graph neural networks In jax. http://github.com/deepmind/jraph
Grattarola D, Alippi C (2020) Graph neural networks in tensorflow and keras with spektral. arXiv . arxiv:2006.12138
Grover, A, Leskovec, J (2016) node2vec: scalable feature learning for networks. arXiv . arxiv:1607.00653
Guillaume J-L, Latapy M (2004) Bipartite structure of all complex networks. Inf Process Lett 90(5):215–221
Hamilton WL (2020) Graph representation learning. Synthesis Lectures Artif Intell Mach Learn 14(3):1–159
Hamilton WL, Ying R, Leskovec J (2017) Representation learning on graphs: methods and applications. arXiv preprint arXiv:1709.05584
Hamilton WL, Ying R, Leskovec J (2018) Representation learning on graphs: methods and applications
Harper FM, Konstan JA (2015) The movielens datasets: history and context. ACM Trans Interact Intell Syst, **5**(4) . https://doi.org/10.1145/2827872

Harper FM, Konstan JA (2015) The movielens datasets: history and context. ACM Trans Interact Intell Syst 5(4):1–19. https://doi.org/10.1145/2827872

Ha D, Dai A, Le QV (2016) HyperNetworks. arXiv. arxiv:1609.09106

He C (2019) Heterogeneous graph convolutional networks for bipartite graph embedding

Hegeman T, Iosup A (2018) Survey of Graph Analysis Applications. arXiv. arxiv:1807.00382

He R, McAuley J (2016) Ups and downs. In: Proceedings of the 25th international conference on world wide web. international world wide web conferences steering committee, ??? . https://doi.org/10.1145/2872427.2883037

He C, Xie T, Rong Y, Huang W, Huang J, Ren X, Shahabi C Adversarial Representation Learning on Large-Scale Bipartite Graphs

He C, Xie T, Rong Y, Huang W, Huang J, Ren X, Shahabi C (2020) Cascade-BGNN: toward efficient self-supervised representation learning on large-scale bipartite graphs

Huang W, Li Y, Fang Y, Fan J, Yang H (2020) Biane: Bipartite attributed network embedding. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, pp 149–158

Jiang M, Cui P, Yuan NJ, Xie X, Yang S (2016) Little is much: Bridging cross-platform behaviors through overlapped crowds. In: Proceedings of the thirtieth AAAI conference on artificial intelligence. AAAI'16, pp 13–19. AAAI Press, ???

Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv . arxiv:1609.02907

Leskovec J, Adamic LA, Huberman BA (2007) The dynamics of viral marketing. ACM Trans Web 1(1):5. https://doi.org/10.1145/1232722.1232727

Leskovec J, Lang KJ, Dasgupta A, Mahoney MW (2008) Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. arXiv . arxiv:0810.1355

Liu Z, Nguyen T-K, Fang Y (2021) Tail-gnn: Tail-node graph neural networks. In: Proceedings of the 27th ACM SIGKDD conference on knowledge discovery and data mining. KDD '21, pp 1109–1119. association for computing machinery, New York, NY, USA . https://doi.org/10.1145/3447548.3467276

Li C, Jia K, Shen D, Shi CJR, Yang H (2019) 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, ??? . https://doi.org/10.24963/ijcai.2019/398

Li C, Shi M, Qu B, Li X (2021) Deep attributed network representation learning via attribute enhanced neighborhood. arXiv . arxiv:2104.05234

Malkov YA, Yashunin DA (2016) Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. arXiv . arxiv:1603.09320

Matthias, F, Jan Eric, L: Fast graph representation learning with pytorch geometric. https://github.com/pyg-team/pytorch_geometric

McCallum A, Nigam K, Rennie JDM, Seymore K (2004) Automating the construction of internet portals with machine learning. Inf Retrieval 3:127–163

McFee B, Bertin-Mahieux T, Ellis DPW, Lanckriet GRG (2012) The million song dataset challenge. In: Proceedings of the 21st international conference on world wide web. WWW '12 Companion, pp 909–916. association for computing machinery, New York, NY, USA . https://doi.org/10.1145/2187980.2188222

Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv . arxiv:1301.3781

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Burges, C.J, Bottou, L, Welling, M, Ghahramani, Z, Weinberger, K.Q (eds.) Advances in neural information processing systems, vol. 26. Curran Associates, Inc., ??? . https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Min S, Gao Z, Peng J, Wang L, Qin K, Fang B (2021) Stgsn - a spatial-temporal graph neural network framework for time-evolving social networks. Knowl-Based Syst 214:106746. https://doi.org/10.1016/j.knosys.2021.106746

Graph Nets, https://www.deepmind.com/open-source/graph-nets

Newman MEJ, Strogatz SH, Watts DJ (2001) Random graphs with arbitrary degree distributions and their applications. Phys Rev E 64(2):026118. https://doi.org/10.1103/physreve.64.026118

Perozzi B, Al-Rfou R, Skiena S (2014) DeepWalk. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data Mining. ACM, ??? . https://doi.org/10.1145/2623330.2623732

Rumelhart DE, Mcclelland J (1986) Parallel distributed processing: explorations in the microstructure of cognition. Vol 1. Foundations,

Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93. https://doi.org/10.1609/aimag.v29i3.2157

Shlens J (2014) Notes on Kullback-Leibler Divergence and Likelihood. arXiv . arxiv:1404.2000

Stauffer M, Tschachtli T, Fischer A, Riesen K (2017). A survey on applications of bipartite graph edit distance. https://doi.org/10.1007/978-3-319-58961-9_22

Sybrandt J, Safro I FOBE and HOBE: First- and High-Order Bipartite Embeddings

Tang J, Zhang J, Yao L, Li J, Zhang L, Su Z (2008) Arnetminer: Extraction and mining of academic social networks. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '08, pp 990–998. association for computing machinery, New York, NY, USA . https://doi.org/10.1145/1401890.1402008

Tang J, Gao H, Liu H, Das Sarma A (2012) Etrust: Understanding trust evolution in an online world. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '12, pp. 253–261. association for computing machinery, New York, NY, USA . https://doi.org/10.1145/2339530.2339574

Tencent records the watching behaviors of users on movies in QQlive. https://v.qq.com/

Wang H, Chen B, Li W-J (2013) Collaborative topic regression with social regularization for tag recommendation. In: IJCAI

Wang M, Zhang J, Liu J, Hu W, Wang S, Li X, Liu W (2017) PDD Graph: bridging electronic medical records and biomedical knowledge graphs via entity linking. arXiv . arxiv:1707.05340

Wang M, Zheng D, Ye Z, Gan Q, Li M, Song X, Zhou J, Ma C, Yu L, Gai Y, Xiao T, He T, Karypis G, Li J, Zhang Z (2019) Deep graph library: a graph-centric, highly-performant package for graph neural networks. arXiv. arxiv:1909.01315

West R, Leskovec J (2012) Human wayfinding in information networks. In: proceedings of the 21st international conference on world wide web. WWW '12, pp 619–628. association for computing machinery, New York, NY, USA . https://doi.org/10.1145/2187836.2187920

West R, Pineau J, Precup D (2009) Wikispeedia: An online game for inferring semantic distances between concepts. In: IJCAI

Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS (2021) A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 32(1):4–24. https://doi.org/10.1109/tnnls.2020.2978386

Xu Y, Cohen SB (2018) Stock movement prediction from tweets and historical prices. In: Proceedings of the 56th annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp 1970–1979. association for computational linguistics, Melbourne, Australia . https://doi.org/10.18653/v1/P18-1183

Yang J, Leskovec J Defining and evaluating network communities based on ground-truth. arXiv arxiv:1205.6233

Yang C, Xiao Y, Zhang Y, Sun Y, Han J: Heterogeneous network representation learning: a unified framework with survey and benchmark. arXiv (2020). arxiv:2004.00216

Yu L, Zhang C, Pei S, Sun G, Zhang X (2018) Walkranker: A unified pairwise ranking model with multiple relations for item recommendation. In: McIlraith, S.A, Weinberger, K.Q (eds.) proceedings of the thirty-second AAAI conference on artificial intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on educational advances in artificial intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pp 2596–2603. AAAI Press, ??? . https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16520

Zhang Y, Xiong Y, Kong X, Zhu Y (2017) Learning node embeddings in interaction graphs. In: Proceedings of the 2017 ACM on conference on information and knowledge management. CIKM '17, pp 397–406. Association for Computing Machinery, New York, NY, USA . https://doi.org/10.1145/3132847.3132918