Unsupervised Attributed Multiplex Network Embedding

Chanyoung Park[1], **Donghyun Kim**[2], **Jiawei Han**[1], **Hwanjo Yu**[3]
[1]Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
[2]Yahoo! Research, CA, USA
[3]Department of Computer Science and Engineering, Pohang University of Science and Technology, Korea
pcy1302@illinois.edu, donghyun.kim@verizonmedia.com, hanj@illinois.edu, hwanjoyu@postech.ac.kr

Abstract

Nodes in a multiplex network are connected by multiple types of relations. However, most existing network embedding methods assume that only a single type of relation exists between nodes. Even for those that consider the multiplexity of a network, they overlook node attributes, resort to node labels for training, and fail to model the global properties of a graph. We present a simple yet effective unsupervised network embedding method for attributed multiplex network called DMGI, inspired by Deep Graph Infomax (DGI) that maximizes the mutual information between local patches of a graph, and the global representation of the entire graph. We devise a systematic way to jointly integrate the node embeddings from multiple graphs by introducing 1) the consensus regularization framework that minimizes the disagreements among the relation-type specific node embeddings, and 2) the universal discriminator that discriminates true samples regardless of the relation types. We also show that the attention mechanism infers the importance of each relation type, and thus can be useful for filtering unnecessary relation types as a preprocessing step. Extensive experiments on various downstream tasks demonstrate that DMGI outperforms the state-of-the-art methods, even though DMGI is fully unsupervised.

1 Introduction

Analyzing and mining useful knowledge in graphs have been an actively researched topic for decades both in academia and industry. Among various graph mining techniques, network embedding, which learns low-dimensional vector representations for nodes in a graph, is shown to be especially effective for various network-based tasks (Tang et al. 2015; Wang et al. 2017; Meng et al. 2019).

However, most existing network embedding methods assume that only a single type of relation exists between nodes (Veličković et al. 2017; 2019; Kipf and Welling 2016), whereas in reality networks are *multiplex* (De Domenico et al. 2013) in nature, i.e., with multiple types of relations. Taking the publication network as an example, two papers can be connected due to various reasons, such as authors (two papers are authored by a common author), citation (one paper cites the other), or keywords (two papers share common keywords). As another example, in a movie database network, two movies can be connected via a common director, or a common actor.

Although different types of relations can independently form different graphs, *these graphs are related*, and thus can mutually help each other for various downstream tasks. As a concrete example of the publication network, although it is hard to infer the topic of a paper only from its citations (citations can be diverse), also knowing other papers written by the same authors will help predict its topic, because authors usually work on a specific research topic. Furthermore, nodes in graphs may contain attribute information, which plays important roles in many applications (Zhang et al. 2018b). For example, if we are additionally given the abstract of the papers in the publication network, it will be much easier to infer their topics. As such, the main challenge is to learn a consensus representation of a node that not only considers its *multiplexity*, but also its *attributes*.

Several recent studies have been conducted for multiplex network embedding, however, some issues remain that need further consideration. First, previous methods (Qu et al. 2017; Zhang et al. 2018a; Shi et al. 2018; Liu et al. 2017) focus on the integration of multiple graphs, but overlook **node attributes**. Second, even for those that consider node attributes (Schlichtkrull et al. 2018; Wang et al. 2019), they require **node labels** for training. However, as node labeling is often expensive and time-consuming, it would be the best if a method can show competitive performance even without any label. Third, most of these methods fail to model the **global properties** of a graph, because they are based on random walk-based skip-gram model or graph convolutional network (GCN) (Kipf and Welling 2016), both of which are known to be effective for capturing the local graph structure (Yadav et al. 2019). More precisely, nodes that are "close" (i.e., within the same context window or neighborhoods) in the graph are trained to have similar representations, whereas nodes that are far apart do not have similar representations, even though they are structurally similar (Ribeiro, Saverese, and Figueiredo 2017).

Keeping these limitations in mind, we propose a simple yet effective unsupervised method for embedding attributed multiplex networks. The core building block of our proposed method is Deep Graph Infomax (DGI) (Veličković et al. 2019) that aims to learn a node encoder that maximizes the mutual information between local patches of a graph,

and the global representation of the entire graph. DGI is the workhorse method for our task, because it 1) naturally integrates the node attributes by using a GCN, 2) is trained in a fully unsupervised manner, and 3) captures the global properties of the entire graph. However, it is challenging to apply DGI, which is designed for embedding a single network, to a multiplex network in which the interactions among multiple relation types, and the importance of each relation type should be considered.

In this paper, we present a systematic way to jointly integrate the embeddings from multiple types of relations between nodes, so as to facilitate them to mutually help each other learn high-quality embeddings useful for various downstream tasks. More precisely, we introduce the *consensus regularization framework* that minimizes the disagreements among the relation-type specific node embeddings, and the *universal discriminator* that discriminates true samples, i.e., ground truth "(graph-level summary, local patch)" pairs, regardless of the relation types. Moreover, we demonstrate that through the *attention mechanism*, we can infer the importance of each relation type in generating the consensus node embeddings, which can be used for filtering unnecessary relation types as a preprocessing step. Our extensive experiments demonstrate that our proposed method, Deep Multplex Graph Infomax (DMGI), outperforms the state-of-the-art attributed multiplex network embedding methods in terms of node clustering, similarity search, and especially, node classification even though DMGI is fully unsupervised.

2 Problem Statement

Definition 1. (Attributed Multiplex Network) An attributed multiplex network is a network $\mathcal{G} = \{\mathcal{G}^1, \mathcal{G}^2, ..., \mathcal{G}^{|\mathcal{R}|}\} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, where $\mathcal{G}^r = \{\mathcal{V}, \mathcal{E}^{(r)}, \mathbf{X}\}$ is a graph of the relation type $r \in \mathcal{R}$, $\mathcal{V}$ is the set of n nodes, $\mathcal{E} = \bigcup_{r \in \mathcal{R}} \mathcal{E}^{(r)} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of all edges with relation type $r \in \mathcal{R}$, and $\mathbf{X} \in \mathbb{R}^{n \times f}$ is a matrix that encodes node attributes information for n nodes. Note that $|\mathcal{R}| > 1$ for multiplex networks, and $|\mathcal{R}| = 1$ for a single network. Given the network $\mathcal{G}$, $\mathcal{A} = \{\mathbf{A}^{(1)}, ..., \mathbf{A}^{(|\mathcal{R}|)}\}$ is a set of adjacency matrices, where $\mathbf{A}^{(r)} \in \{0, 1\}^{|V| \times |V|}$ is an adjacency matrix of the network $\mathcal{G}^r$.

Task: Unsupervised Attributed Multiplex Network Embedding. Given an attributed multiplex network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, and the set of adjacency matrices $\mathcal{A}$, the task of unsupervised attributed multiplex network embedding is to learn a d-dimensional vector representation $\mathbf{z}_i \in \mathbf{Z} \in \mathbb{R}^{n \times d}$ for each node $v_i \in \mathcal{V}$ without using any labels.

3 Unsupervised Attributed Multiplex Network Embedding

We begin by introducing Deep Graph Informax (DGI) (Veličković et al. 2019), then we discuss about its limitations, and present our proposed method.

Deep Graph Infomax (DGI). Veličković et al. (2019) proposed an unsupervised method for learning node representations, called DGI, that relies on the infomax principle (Linsker 1988). More precisely, DGI aims to learn a low-dimensional vector representation for each node v_i, i.e., $\mathbf{h}_i \in \mathbb{R}^d$, such that the average mutual information (MI) between the graph-level (global) summary representation $\mathbf{s} \in \mathbb{R}^d$, and the representations of the local patches $\{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n\}$ is maximized. To this end, DGI introduces a discriminator $\mathcal{D}$ that discriminates the true samples, i.e., $(\mathbf{h}_i, \mathbf{s})$, from its negative counterparts, i.e., $(\tilde{\mathbf{h}}_j, \mathbf{s})$:

$$\mathcal{L} = \sum_{v_i \in \mathcal{V}}^{n} \log \mathcal{D}\left(\mathbf{h}_i, \mathbf{s}\right) + \sum_{j=1}^{n} \log\left(1 - \mathcal{D}\left(\tilde{\mathbf{h}}_j, \mathbf{s}\right)\right) \quad (1)$$

where $\mathbf{h}_i = \sigma\left(\sum_{j \in N(i)} \frac{1}{c_{ij}} \mathbf{x}_j \mathbf{W}\right)$, $N(i)$ is the set of neighboring nodes of v_i including v_i itself, $\mathbf{W} \in \mathbb{R}^{f \times d}$, and c_{ij} is a normalizing constant for edge (v_i, v_j), $\mathbf{s} = \sigma\left(\frac{1}{n} \sum_{i=1}^{n} \mathbf{h}_i\right)$, and σ is the sigmoid nonlinearity. Negative patch representation $\tilde{\mathbf{h}}_j$ is obtained by row-wise shuffling the original attribute matrix $\mathbf{X}$. Veličković et al. (2019) theoretically proved that the binary cross entropy loss shown in Eqn. 1 amounts to maximizing the mutual information (MI) between $\mathbf{h}_i$ and $\mathbf{s}$, based on the Jensen-Shannon divergence (Veličković et al. 2019). Refer to Section 3.3 of (Veličković et al. 2019) for the detailed proof. As the local patch representations $\{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n\}$ are learned to preserve the MI with the graph-level representation $\mathbf{s}$, each $\mathbf{h}_i$ is expected to capture the global properties of the entire graph.

Limitation. Despite its effectiveness, DGI is designed for a single attributed network, and thus it is not straightforward to apply it to a multiplex network. As a naive extension of DGI to a multiplex attributed network, we can independently apply DGI to each graph formed by each relation type, and then compute the average of the embeddings obtained from each graph to get the final node representations. However, we argue that this fails to model the multiplexity of the network, because the interactions among the node embeddings from different relation types is not captured. Thus, we need a more systematic way to integrate multiple independent models to obtain the final consensus embedding that every model can agree on.

3.1 Deep Multiplex Graph Infomax: DMGI

We present our unsupervised method for embedding an attributed multiplex network. We first describe how to independently model each graph pertaining to each relation type, then explain how to jointly integrate them to finally obtain the consensus node embedding matrix.

Relation-type specific Node Embedding. For each relation type $r \in \mathcal{R}$, we introduce a relation-type specific node encoder $g_r : \mathbb{R}^{n \times f} \times \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times d}$ to generate the relation-type specific node embedding matrix $\mathbf{H}^{(r)}$ of nodes in $\mathcal{G}^{(r)}$. The encoder is a single–layered GCN:

$$\mathbf{H}^{(r)} = g_r(\mathbf{X}, \mathbf{A}^{(r)} | \mathbf{W}^{(r)}) = \sigma\left(\hat{\mathbf{D}}_r^{-\frac{1}{2}} \hat{\mathbf{A}}^{(r)} \hat{\mathbf{D}}_r^{-\frac{1}{2}} \mathbf{X} \mathbf{W}^{(r)}\right) \quad (2)$$

where $\hat{\mathbf{A}}^{(r)} = \mathbf{A}^{(r)} + w\mathbf{I}_n$, $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$, $\mathbf{W}^{(r)} \in \mathbb{R}^{f \times d}$ is a trainable weight matrix of the relation-type specific decoder g_r, and σ is the ReLU nonlinearity. Unlike conventional GCNs (Kipf and Welling 2016), we control the weight

of the self-connections by introducing a weight $w \in \mathbb{R}$. Larger w indicates that the node itself plays a more important role in generating its embedding, which in turn diminishes the importance of its neighboring nodes. Then, we compute the graph-level summary representation $\mathbf{s}^{(r)}$ that summarizes the global content of the graph $\mathcal{G}^{(r)}$. We employ a readout function $\mathsf{Readout} : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{d}$:

$$\mathbf{s}^{(r)} = \mathsf{Readout}(\mathbf{H}^{(r)}) = \sigma\left(\frac{1}{n}\sum_{i=1}^{n}\mathbf{h}_i^{(r)}\right) \tag{3}$$

where σ is the logistic sigmoid nonlinearity, and $\mathbf{h}_i^{(r)}$ denotes the i-th row vector of the matrix $\mathbf{H}^{(r)}$. We also note that various pooling methods such as maxpool, and SAGPool (Lee, Lee, and Kang 2019) can be used as $\mathsf{Readout}(\cdot)$.

Next, given the relation-type specific node embedding matrix $\mathbf{H}^{(r)}$, and its graph-level summary representation $\mathbf{s}^{(r)}$, we compute the relation-type specific cross entropy:

$$\mathcal{L}^{(r)} = \sum_{v_i \in \mathcal{V}}^{n} \log \mathcal{D}\left(\mathbf{h}_i^{(r)}, \mathbf{s}^{(r)}\right) + \sum_{j=1}^{n} \log\left(1 - \mathcal{D}\left(\tilde{\mathbf{h}}_j^{(r)}, \mathbf{s}^{(r)}\right)\right) \tag{4}$$

where $\mathcal{D} : \mathbb{R}^{d} \times \mathbb{R}^{d} \rightarrow \mathbb{R}$ is a discriminator that scores patch-summary representation pairs, i.e., $(\mathbf{h}_i^{(r)}, \mathbf{s}^{(r)})$. In this paper, we apply a simple bilinear scoring function as it empirically performs the best in our experiments:

$$\mathcal{D}\left(\mathbf{h}_i^{(r)}, \mathbf{s}^{(r)}\right) = \sigma(\mathbf{h}_i^{(r)T}\mathbf{M}^{(r)}\mathbf{s}^{(r)}) \tag{5}$$

where σ is the logistic sigmoid nonlinearity, and $\mathbf{M}^{(r)} \in \mathbb{R}^{d \times d}$ is a trainable scoring matrix. To generate the negative node embedding $\tilde{\mathbf{h}}_j^{(r)}$, we corrupt the original attribute matrix by shuffling it in the row-wise manner (Veličković et al. 2019), i.e., $\tilde{\mathbf{X}} \leftarrow \mathbf{X}$, and reuse the encoder in Eqn. 2. i.e. $\tilde{\mathbf{H}}^{(r)} = g_r(\tilde{\mathbf{X}}, \mathbf{A}^{(r)}|\mathbf{W}^{(r)})$.

Joint Modeling and Consensus Regularization. Heretofore, by independently maximizing the average MI between the local patches $\{\mathbf{h}_1^{(r)}, \mathbf{h}_2^{(r)}, ..., \mathbf{h}_n^{(r)}\}$ and the graph-level summary $\mathbf{s}^{(r)}$ pertaining to each graph $\mathcal{G}^{(r)}(\forall r \in \mathcal{R})$, we obtained relation-type specific node embedding matrix $\mathbf{H}^{(r)}$ that captures the global information in $\mathcal{G}^{(r)}$. However, as each $\mathbf{H}^{(r)}$ is trained independently for each $r \in \mathcal{R}$, these embedding matrices only contain relevant information regarding each relation type, and therefore fail to take advantage of the multiplexity of the network. This motivates us to develop a systematic way to jointly integrate the embeddings from different relation types, so as to facilitate them to mutually help each other learn high-quality embeddings.

To this end, we introduce the consensus embedding matrix $\mathbf{Z} \in \mathbb{R}^{n \times d}$ on which every relation-type specific node embedding matrix $\mathbf{H}^{(r)}$ can agree. More precisely, we introduce the *consensus regularization* framework that consists of 1) a regularizer minimizing the disagreements between the set of original node embeddings, i.e. $\{\mathbf{H}^{(r)} \mid r \in \mathcal{R}\}$ and the consensus embedding $\mathbf{Z}$, and 2) another regularizer maximizing the disagreement between the corrupted node

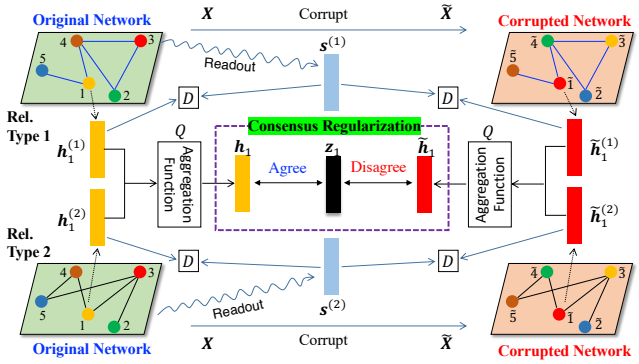

Figure 1: Overview of DMGI (Best viewed in color).

embeddings, i.e., $\{\tilde{\mathbf{H}}^{(r)} \mid r \in \mathcal{R}\}$, and the consensus embedding $\mathbf{Z}$, which are formulated as follows:

$$\ell_{\text{cs}} = \left[\mathbf{Z} - \mathcal{Q}\left(\{\mathbf{H}^{(r)} \mid r \in \mathcal{R}\}\right)\right]^2 - \left[\mathbf{Z} - \mathcal{Q}\left(\{\tilde{\mathbf{H}}^{(r)} \mid r \in \mathcal{R}\}\right)\right]^2 \tag{6}$$

where $\mathcal{Q}$ is an aggregation function that combines a set of node embedding matrices from multiple relation types into a single embedding matrix. i.e., $\mathbf{H} \in \mathbb{R}^{n \times d}$. $\mathcal{Q}$ can be any pooling method that can handle permutation invariant input, such as set2set (Vinyals, Bengio, and Kudlur 2015) or Set Transformer (Lee et al. 2019). However, considering the efficiency of the method, we simply employ average pooling, i.e., computing the average of the set of embedding matrices:

$$\mathbf{H} = \mathcal{Q}\left(\{\mathbf{H}^{(r)} \mid r \in \mathcal{R}\}\right) = \frac{1}{|\mathcal{R}|}\sum_{r \in \mathcal{R}}\mathbf{H}^{(r)} \tag{7}$$

It is important to note that the scoring matrix $\mathbf{M}^{(*)}$ in Eqn. 5 is shared among all the relations $r \in \mathcal{R}$. i.e., $\mathbf{M} = \mathbf{M}^{(1)} = \mathbf{M}^{(2)} = ... = \mathbf{M}^{(|\mathcal{R}|)}$. The intuition is to learn the *universal discriminator* that is capable of scoring the true pairs higher than the negative pairs regardless the relation types. We argue that the universal discriminator facilitates the joint modeling of different relation types together with the consensus regularization.

Finally, we jointly optimize the sum of all the relation-type specific loss in Eqn. 4, and the consensus regularization in Eqn. 6 to obtain the final objective $\mathcal{J}$ as follows:

$$\mathcal{J} = \sum_{r \in \mathcal{R}} \mathcal{L}^{(r)} + \alpha\ell_{\text{cs}} + \beta||\Theta||^2 \tag{8}$$

where α controls the importance of the consensus regularization, β is a coefficient for l2 regularization on Θ, which is a set of trainable parameters. i.e., $\Theta = \{\{\mathbf{W}^{(r)} \mid r \in \mathcal{R}\}, \mathbf{M}, \mathbf{Z}\}$, and $\mathcal{J}$ is optimized by Adam optimizer. Figure 1 illustrates the overview of DMGI.

Discussion. Despite its efficiency, the above average pooling scheme in Eqn. 7 treats all the relations equally, whereas, as will be shown in the experiments, some relation type is more beneficial for a certain downstream task than others. For example, the co-authorship information between two papers plays a more significant role in predicting the topic of a

Table 1: Statistics of the datasets. The node attributes are bag-of-words of text associated with each node.

	Relations (A-B)	Num. A	Num. B	Num. A-B	Relation type	Num. relations	Num. node attributes	Num. labeled data	Num. classes
ACM	Paper-Author	3,025	5,835	9,744	P-A-P	29,281	1,830 (Paper abstract)	600	3
	Paper-Subject	3,025	56	3,025	P-S-P	2,210,761			
IMDB	Movie-Actor	3,550	4,441	10,650	M-A-M	66,428	1,007 (Movie plot)	300	3
	Movie-Director	3,550	1,726	3,550	M-D-M	13,788			
DBLP	Paper-Author	7,907	1,960	14,238	P-A-P	144,783	2,000 (Paper abstract)	80	4
	Paper-Paper	7,907	7,907	10,522	P-P-P	90,145			
	Author-Term	1,960	1,975	57,269	P-A-T-A-P	57,137,515			
Amazon	Item-Item	7,621	7,621	38,514	Also-view	266,237	2,000 (Item description)	80	4
				45,446	Also-bought	1,104,257			
				9,783	Bought-together	16,305			

paper compared with their citation information; eventually, these two information mutually help each other to more accurately predict the topic of a paper. Therefore, we can adopt the attention mechanism (Bahdanau, Cho, and Bengio 2014) to distinguish between different relation types as follows:

$$\mathbf{h}_i = \mathcal{Q}\left(\{\mathbf{h}^{(r)} \mid r \in \mathcal{R}\}\right) = \sum_{r\in\mathcal{R}} a_i^{(r)}\mathbf{h}^{(r)} \tag{9}$$

where $a_i^{(r)}$ denotes the importance of relation r in generating the final embedding of node v_i defined as:

$$a_i^{(r)} = \frac{\exp\left(\mathbf{q}^{(r)} \cdot \mathbf{h}_i^{(r)}\right)}{\sum_{r'\in\mathcal{R}} \exp\left(\mathbf{q}^{(r')} \cdot \mathbf{h}_i^{r'}\right)} \tag{10}$$

where $\mathbf{q}^{(r)} \in \mathbb{R}^d$ is the feature vector of relation r.

Extension to Semi-Supervised Learning. It is important to note that DMGI is trained in a *fully unsupervised* manner. However, in reality, nodes are sometimes associated with label information, which can guide the training of node embeddings even with a small amount (Kipf and Welling 2016; Qu et al. 2017). To this end, we introduce a *semi-supervised module* into our framework that predicts the labels of labeled nodes from the consensus embedding $\mathbf{Z}$. More precisely, we minimize the cross-entropy error over the labeled nodes:

$$\ell_{\text{sup}} = -\frac{1}{|\mathcal{Y}_L|}\sum_{l\in\mathcal{Y}_L}\sum_{i=1}^{c} Y_{li}\ln\hat{Y}_{li} \tag{11}$$

where $\mathcal{Y}_L$ is the set of node indices with labels, $Y \in \mathbb{R}^{n\times c}$ is the ground truth label, $\hat{Y} = \mathsf{softmax}(f(\mathbf{Z}))$ is the output of a softmax layer, and $f : \mathbb{R}^{n\times d} \rightarrow \mathbb{R}^{n\times c}$ is a classifier that predicts the label of a node from its embedding, which is a single fully connected layer in this work. The final objective function with the semi-supervised module is:

$$\mathcal{J}_{\mathsf{semi}} = \sum_{r\in\mathcal{R}} \mathcal{L}^{(r)} + \alpha\ell_{\text{cs}} + \beta||\Theta|| + \gamma\ell_{\text{sup}} \tag{12}$$

where γ the coefficient of the semi-supervised module.

4 Experiments

Dataset. To make fair comparisons with HAN (Wang et al. 2019), which is the most relevant baseline method, we evaluate our proposed method on the datasets used in their original paper (Wang et al. 2019), i.e., ACM, DBLP, and IMDB. We used publicly available ACM dataset (Wang et al. 2019), and preprocessed DBLP and IMDB datasets. For ACM and DBLP datasets, the task is to classify the papers into three classes (Database, Wireless Communication, Data Mining), and four classes (DM, AI, CV, NLP)[1], respectively, according to the research topic. For IMDB dataset, the task is to classify the movies into three classes (Action, Comedy, Drama). We note that the above datasets used by previous work are not truly multiplex in nature because the multiplexity between nodes is inferred via intermediate nodes (e.g., ACM: Paper-Paper relationships are inferred via Authors and Subjects that connect two Papers. i.e., "PAP" and "PSP"). Thus, to make our evaluation more practical, we used Amazon dataset (He and McAuley 2016) that genuinely contains a multiplex network of items, i.e., also-viewed, also-bought, and bought-together relations between items. We used datasets from four categories[2], i.e., Beauty, Automotive, Patio Lawn and Garden, and Baby, and the task is to classify items into the four classes. For ACM and IMDB datasets, we used the same number of labeled data as in (Wang et al. 2019) for fair comparisons, and for the remaining datasets, we used 20 labeled data for each class. Table 1 summarizes the data statistics.

Methods Compared.

1) Embedding methods for a single network

- No attributes: **Deepwalk** (Perozzi, Al-Rfou, and Skiena 2014), **node2vec** (Grover and Leskovec 2016): They learn node embeddings by random walks and skip-gram.
- Attributed network embedding: **GCN** (Kipf and Welling 2016), **GAT** (Veličković et al. 2017): They learn node embeddings based on local neighborhood structures. As they perform similarly, we report the best performing method among them; **DGI** (Veličković et al. 2019): It maximizes the MI between the graph-level summary representation and the local patches; **ANRL** (Zhang et al. 2018b): It uses neighbor enhancement autoencoder to model the node attribute information, and skip-gram model to capture the network structure; **CAN** (Meng et al. 2019): It learns embeddings of both attributes and nodes in the same semantic

[1] **DM**: KDD,WSDM,ICDM, **AI**: ICML,AAAI,IJCAI, **CV**: CVPR, **NLP**: ACL,NAACL,EMNLP

[2] We chose these categories because the three types of item-item relations from these categories are similar in number

Table 2: Properties of the compared methods (*Mult.*: Mutliplexity, *Attr*: Attribute, *Unsup*: Unsupervised, *Glo*: Global).

	Mult.	*Attr.*	*Unsup.*	*Glo.*
Dw/n2v	✗	✗	✓	✗
GCN/GAT	✗	✓	✗	✗
DGI	✗	✓	✓	✗
ANRL	✗	✓	✓	✓
CAN	✗	✓	✓	✗
DGCN	✗	✓	✗	✓
CMNA	✓	✗	✓	✓
MNE	✓	✗	✓	✗
mGCN	✓	✗	✓	✗
HAN	✓	✓	✗	✗
DMGI	✓	✓	✓	✓

Table 3: Performance for node clustering and similarity search on test data.

Method	ACM		IMDB		DBLP		Amazon	
	NMI	Sim@5	NMI	Sim@5	NMI	Sim@5	NMI	Sim@5
Deepwalk	0.310	0.710	0.117	0.490	0.348	0.629	0.083	0.726
node2vec	0.309	0.710	0.123	0.487	0.382	0.629	0.074	0.738
GCN/GAT	0.671	0.867	0.176	0.565	0.465	0.724	0.287	0.624
DGI	0.640	0.889	0.182	0.578	0.551	0.786	0.007	0.558
ANRL	0.515	0.814	0.163	0.527	0.332	0.720	0.166	0.763
CAN	0.504	0.836	0.074	0.544	0.323	0.792	0.001	0.537
DGCN	0.691	0.690	0.143	0.179	0.462	0.491	0.143	0.194
CMNA	0.498	0.363	0.152	0.069	0.420	0.511	0.070	0.435
MNE	0.545	0.791	0.013	0.482	0.136	0.711	0.001	0.395
mGCN	0.668	0.873	0.183	0.550	0.468	0.726	0.301	0.630
HAN	0.658	0.872	0.164	0.561	0.472	0.779	0.029	0.495
DMGI	0.687	0.898	**0.196**	**0.605**	0.409	0.766	**0.425**	0.816
DMGI$_{attn}$	**0.702**	**0.901**	0.185	0.586	**0.554**	**0.798**	0.412	**0.825**

space; **DGCN** (Zhuang and Ma 2018): It models the local and global properties of a graph by employing dual GCNs.

2) Multiplex embedding methods

- No attributes: **CMNA** (Chu et al. 2019): It leverages the cross-network information to refine inter-vector for network alignment and intra-vector for other downstream tasks. We use the intra-vector for our evaluations; **MNE** (Zhang et al. 2018a): It jointly models multiple networks by introducing a common embedding, and a additional embedding for each relation type.
- Attributed multiplex network embedding: **mGCN** (Ma et al. 2019), **HAN** (Wang et al. 2019): They apply GCNs, and GATs on multiplex network considering the inter-, and intra-network interactions. For fair comparisons, we initialized the initial node embeddings of mGCN by using the node attribute matrix, although the node attributes information is ignored in the original mGCN; **DMGI$_{attn}$**: DMGI with the attention mechanism (Eqn. 9).

For the sake of fair comparisons with DMGI, which considers the node attributes, we concatenated the raw attribute matrix $\mathbf{X}$ to the learned node embeddings $\mathbf{Z}$ of the methods that ignore the node attributes. i.e., Deepwalk, node2vec, CMNA, and MNE. i.e., $\mathbf{Z} \leftarrow [\mathbf{Z}; \mathbf{X}]$. Moreover, regarding the embedding methods for a single network, i.e., the methods that belong to the first category in the above list, we obtain the final node embedding matrix $\mathbf{Z}$ by computing the average of the node embeddings obtained from each single graph. i.e., $\mathbf{Z} = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} \mathbf{H}^{(r)}$. We provide a summary of the properties of the compared methods in Table 2.

Evaluation Metrics. Recall that DMGI is an unsupervised method that does not require any labeled data for training. Therefore, we evaluate the performance of DMGI in terms of **node clustering** and **similarity search**, both of which are classical performance measures for unsupervised methods. For node clustering, we use the most commonly used metric (Wang et al. 2019), i.e., Normalized Mutual Information (NMI). For similarity search, we compute the cosine similarity scores of the node embeddings between all pairs of nodes, and for each node, we rank the nodes according to the similarity score. Then, we calculate the ratio of the nodes that belong to the same class within top-5 ranked nodes (Sim@5). Moreover, we also evaluate DMGI on the performance in terms of **node classification**. More precisely, after learning the node embeddings, we train a logistic regression classifier on the learned embeddings in the training set, and then evaluate on the nodes in the test set. We use Macro-F1 (MaF1) and Micro-F1 (MiF1) (Wang et al. 2019).

Experimental Settings. We randomly split our dataset into train/validation/test, and we have the equal number of labeled data for training and validation datasets. We report the test performance when the performance on validation data gives the best result. For DMGI, we set the node embedding dimension $d = 64$, self-connection weight $w = 3$, tune $\alpha, \beta, \gamma \in \{0.0001, 0.001, 0.01, 0.1\}$. We implement DMGI in PyTorch[3], and for all other methods, we used the source codes published by the authors, and tried to tune them to their best performance. More precisely, apart from the guidelines provided by the original papers, we tuned learning rate, and the coefficients for regularization from $\{0.0001, 0.0005, 0.001, 0.005\}$ on the validation dataset. After learning the node embeddings, for fair comparisons, we conducted the evaluations within the same platform.

4.1 Performance Analysis

Overall evaluation. Table 3 and Table 4 show the evaluation results on unsupervised and supervised task, respectively. We have the following observations: 1) Our proposed DMGI and DMGI$_{attn}$ outperform all the state-of-the-art baselines not only on the unsupervised tasks, but also the supervised task, although the improvement is more significant in the unsupervised task as expected. This verifies the benefit of our framework that models the multiplexity and the global property of a network together with the node attributes within a single framework. 2) Although DGI shows relatively good performance, the performance is unstable (poor performance on Amazon dataset), indicating that multiple relation types should be jointly modeled. 3) Attribute-aware multiplex network embedding methods,

[3] https://github.com/pcy1302/DMGI

Table 4: Node classification performance on test data.

	ACM		IMDB		DBLP		Amazon	
	MaF1	MiF1	MaF1	MiF1	MaF1	MiF1	MaF1	MiF1
Deepwalk	0.739	0.748	0.532	0.550	0.533	0.537	0.663	0.671
node2vec	0.741	0.749	0.533	0.550	0.543	0.547	0.662	0.669
GCN/GAT	0.869	0.870	0.603	0.611	0.734	0.717	0.646	0.649
DGI	0.881	0.881	0.598	0.606	0.723	0.720	0.403	0.418
ANRL	0.819	0.820	0.573	0.576	0.770	0.699	0.692	0.690
CAN	0.590	0.636	0.577	0.588	0.702	0.694	0.498	0.499
DGCN	0.888	0.888	0.582	0.592	0.707	0.698	0.478	0.509
CMNA	0.782	0.788	0.549	0.566	0.566	0.561	0.657	0.665
MNE	0.792	0.797	0.552	0.574	0.566	0.562	0.556	0.567
mGCN	0.858	0.860	0.623	0.630	0.725	0.713	0.660	0.661
HAN	0.878	0.879	0.599	0.607	0.716	0.708	0.501	0.509
DMGI	**0.898**	**0.898**	**0.648**	**0.648**	0.771	0.766	0.746	0.748
DMGI$_{attn}$	0.887	0.887	0.602	0.606	**0.778**	**0.770**	**0.758**	**0.758**

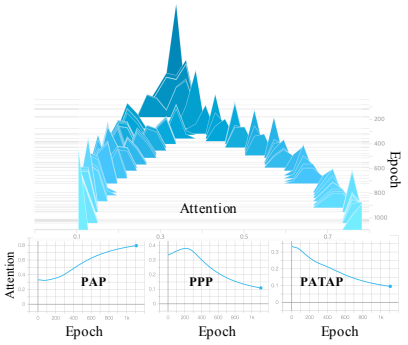

Figure 2: Visualization of the attention weights on DBLP dataset.

Table 5: Performance of similarity search (Sim@5) of embedding methods for a single network. (Merged denotes the average of all the relation-type specific embeddings.)

ACM		GCN	DGI	ANRL	DMGI	DMGI$_{attn}$
Rel. Type	PAP	0.822	0.875	0.795		
	PSP	0.721	0.675	0.694		
Merged		0.867	0.889	0.814	0.898	**0.901**
IMDB		GCN	DGI	ANRL	DMGI	DMGI$_{attn}$
Rel. Type	MAM	0.485	0.484	0.495		
	MDM	0.548	0.562	0.520		
Merged		0.566	0.578	0.527	**0.605**	0.586
DBLP		GCN	DGI	ANRL	DMGI	DMGI$_{attn}$
Rel. Type	PAP	0.730	0.779	0.692		
	PPP	0.456	0.477	0.680		
	PATAP	0.431	0.409	OOM		
Merged		0.724	0.786	0.720	0.766	**0.799**
Amazon		GCN	DGI	ANRL	DMGI	DMGI$_{attn}$
Rel. Type	Also-V	0.355	0.367	0.563		
	Also-B	0.357	0.381	0.516		
	Bou.-T	0.662	0.639	0.770		
Merged		0.624	0.558	0.764	0.816	**0.825**

such as mGCN and HAN, generally perform better than those that neglect the node attributes. i.e., CMNA and MNE, even though we concatenated node attributes to the node embeddings. This verifies not only the benefit of modeling the node attributes, but also that the attributes should be systematically incorporated into the model. 4) Multiplex network embedding methods generally outperform single network embedding methods, although the gap is not significant. This verifies that the multiplexity of a network should be carefully modeled, otherwise a simple aggregation of multiple relation-type specific embeddings learned from independent single network embedding methods may perform better.

Effect of the attention mechanism. In Table 5, we show the performance of DMGI and DMGI$_{attn}$, together with the performance of single network embedding methods (GCN/GAT, DGI, and ANRL). We observe that DMGI$_{attn}$ outperforms DMGI in most of the datasets but IMDB dataset. To analyze the reason for this, we first plot the distribution of the attention weights on DBLP dataset over the training epochs in Figure 2. The above graph in Figure 2 demonstrates that the attention weights eventually end up in both extremes. i.e., close to 0 or close to 1, and the below graphs show that most of the attention weight is dedicated to a single relation type, i.e., "PAP", which actually turns out to be the most important relation among the three (See Table 5); This phenomenon is common in every dataset. Next, we look at the performance of the single network embedding methods, especially DGI, on each relation type in Table 5. We observe that the performance differences among relation types in ACM, DBLP, and Amazon datasets are more biased to a single relation type, whereas in IMDB dataset, "MAM" and "MDM" relations relatively show similar performance. To summarize our findings, since the attention mechanism tends to favor the single most important relation type ("PAP" in ACM, "MDM" in IMDB, "PAP" in DBLP, and "Bought-together" in Amazon), DMGI$_{attn}$ outperforms DMGI on datasets where one relation type significantly outperforms the other, i.e., ACM, DBLP, and Amazon, by removing the noise from other relations. On the other hand, for datasets where all the relations show relatively even performance, i.e., IMDB, extremely favoring a single well performing relation type ("MDM") is rather detrimental to the overall performance because the relation "MAM" should also be considered to some extent.

We also note that since the attention mechanism of DMGI$_{attn}$ can infer the importance of each relation type, we can filter out unnecessary relation types as a preprocessing step. To verify this, we evaluated on all possible combinations of relation types in DBLP dataset (Table 6). We observe that by removing the relation "PATAP", which turned out to be the most useless relation type in Table 5, DMGI$_{attn}$ obtains even better results than using all the relation types, whereas for GCN and DGI, still considering all the relation types shows the best performance. This indicates that the attention mechanism can be useful to filter out unnecessary relation types, which will especially come in handy when the number of relation types is large.

Ablation study. To measure the impact of each component of DMGI$_{attn}$, we conduct ablation studies on the largest dataset, i.e., DBLP, in Table 7. We have the following observations: 1) As expected, the semi-supervised module specif-

Table 6: NMI on various combinations of relation types.

DBLP dataset		GCN/GAT	DGI	DMGI$_{attn}$
NMI	PAP+PPP	0.464	0.543	**0.565**
	PAP+PATAP	0.458	0.535	0.017
	PPP+PATAP	0.332	0.237	0.201
	All	**0.465**	**0.551**	0.554

Table 7: Result for ablation studies of DMGI$_{attn}$.

DBLP dataset		MaF1	NMI	Sim@5
DMGI$_{attn}$		0.778	0.554	0.798
1) DMGI$_{attn}$+ Semi supervised		0.791	0.555	0.798
2) Readout (Eqn. 3)	Random sample	0.774	0.555	0.797
	Maxpool	0.778	0.552	0.802
	Linear projection	0.783	0.565	0.803
	SAGPool	0.797	0.563	0.797
3) Without 2nd term of Eqn. 6		0.749	0.448	0.787
4) $\mathbf{M} \neq \mathbf{M}^{(1)} \neq ... \neq \mathbf{M}^{(\lvert\mathcal{R}\rvert)}$.		0.645	0.076	0.677
5) No attributes (Adj. as attribute)		0.377	0.053	0.763
6) Neg sample: Shuffle adj.		0.364	0.156	0.504

ically helps improve the node classification performance, which is a supervised task, whereas the performance on the unsupervised task remains on par. 2) Various readout functions including ones that contain trainable weights (Linear projection and SAGPool (Lee, Lee, and Kang 2019)) do not have much impact on the performance, which promotes our use of average pooling. 3) The second term in Eqn. 6 indeed plays a significant role in the consensus regularization framework. 4) The sharing of the scoring matrix $\mathbf{M}$ facilitates DMGI to model the interaction among multiple relation types. 5) Node attributes are crucial for representation learning of nodes. 6) Shuffling adjacency matrix instead of attribute matrix deteriorates the model performance.

5 Related Work

Network embedding. Network embedding methods aim at learning low-dimensional vector representation for nodes in a graph while preserving the network structure (Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016; Tang et al. 2015), and various other properties such as node attributes (Zhang et al. 2018b; Meng et al. 2019), structural role (Ribeiro, Saverese, and Figueiredo 2017), and node label information (Huang, Li, and Hu 2017).

Multiplex Network embedding. A multiplex network, which is also known as a multi-view network (Tang et al. 2015; Shi et al. 2018) or a multi-dimensional network (Ma et al. 2018; 2019) in the literature, consists of multiple relation types among a set of single-typed nodes. It can be thought of as a special type of heterogeneous network (Dong, Chawla, and Swami 2017; Fu, Lee, and Lei 2017) with a single type of node and multiple types of edges. Therefore, a multiplex network calls for a special attention because there is no need to consider the semantics between different types of nodes, which is often addressed by the concept of metapath (Sun et al. 2011). Distinguished from heterogeneous network, a key challenge in the multiplex network embedding is to learn a consensus embedding for each node by taking into account the interrelationship among the multiple graphs. In this regard, existing methods mainly focused on how to integrate the information from multiple graphs. HAN (Wang et al. 2019) employed graph attention network (Veličković et al. 2017) on each graph, and then applied the attention mechanism to merge the node representations learned from each graph by considering the importance of each graph. However, the existing methods either require labels for training (Wang et al. 2019; Qu et al. 2017; Schlichtkrull et al. 2018), or overlook the node attributes (Liu et al. 2017; Xu et al. 2017; Li et al. 2018; Shi et al. 2018; Zhang et al. 2018a; Ni et al. 2018; Chu et al. 2019). Most recently, Ma et al. (2019) proposed a graph convolutional network (GCN) based method called mGCN, which is not only unsupervised, but also naturally incorporates the node attributes by using GCNs. However, since it is based on GCNs that capture the local graph structure (Yadav et al. 2019), it fails to fully model the global properties of a graph (Zhuang and Ma 2018; Wang, Cui, and Zhu 2016; Veličković et al. 2019).

Attributed Network Embedding. Nodes in a network are often affiliated with various contents, such as abstract text in the publication network, user profiles in social networks, and item description text in movie database or item networks. Such networks are called attributed networks, and have been extensively studied (Li et al. 2017; Hamilton, Ying, and Leskovec 2017; Yang et al. 2015; Zhang et al. 2018b; Gao and Huang 2018; Zhou et al. 2018; Veličković et al. 2019; Meng et al. 2019). Their goal is to preserve not only the network structure, but also the node attribute proximity in learning representations. Recently, GCNs (Kipf and Welling 2016; Veličković et al. 2017; 2019) have been widely praised for its seamless integration of the network structure, and node attributes into a single framework.

Mutual Information. it has been recently made possible to compute the MI between high dimensional input/output pairs of deep neural networks (Belghazi et al. 2018). Several recent work adopted the infomax principle (Linsker 1988) to learn the unsupervised representations in different domains, such as images (Hjelm et al. 2019), speech (Ravanelli and Bengio 2018) and graphs (Veličković et al. 2019). More precisely, Veličković et al. (2019) proposed Deep Graph Infomax (DGI) for learning representations of graph structured inputs by maximizing the MI between a high-level global representation, and the local patches of a graph.

6 Conclusion

We presented a simple yet effective unsupervised method for embedding attributed multiplex network. DMGI can jointly integrate the embeddings from multiple types of relations between nodes through the consensus regularization framework, and the universal discriminator. Moreover, the attention mechanism of DMGI$_{attn}$ can infer the importance of each relation type, which facilitates the preprocessing of the multiplex network. Experimental results on not only unsupervised tasks, but also a supervised task verify the superiority of our proposed framework.

Acknowledgment: IITP2018-0-00584, IITP-2019-2011-1-00783, 2016R1E1A1A01942642, 2017M3C4A7063570.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Belghazi, M. I.; Baratin, A.; Rajeswar, S.; Ozair, S.; Bengio, Y.; Courville, A.; and Hjelm, R. D. 2018. Mine: mutual information neural estimation. *ICML*.

Chu, X.; Fan, X.; Yao, D.; Zhu, Z.; Huang, J.; and Bi, J. 2019. Cross-network embedding for multi-network alignment. In *WWW*.

De Domenico, M.; Solé-Ribalta, A.; Cozzo, E.; Kivelä, M.; Moreno, Y.; Porter, M. A.; Gómez, S.; and Arenas, A. 2013. Mathematical formulation of multilayer networks. *Physical Review X*.

Dong, Y.; Chawla, N. V.; and Swami, A. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*. ACM.

Fu, T.-y.; Lee, W.-C.; and Lei, Z. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *CIKM*. ACM.

Gao, H., and Huang, H. 2018. Deep attributed network embedding. In *IJCAI*.

Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *KDD*. ACM.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NIPS*.

He, R., and McAuley, J. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*.

Hjelm, R. D.; Fedorov, A.; Lavoie-Marchildon, S.; Grewal, K.; Trischler, A.; and Bengio, Y. 2019. Learning deep representations by mutual information estimation and maximization. *ICLR*.

Huang, X.; Li, J.; and Hu, X. 2017. Label informed attributed network embedding. In *WSDM*. ACM.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *ICLR*.

Lee, J.; Lee, Y.; Kim, J.; Kosiorek, A. R.; Choi, S.; and Teh, Y. W. 2019. Set transformer. *ICML*.

Lee, J.; Lee, I.; and Kang, J. 2019. Self-attention graph pooling. *ICML*.

Li, J.; Dani, H.; Hu, X.; Tang, J.; Chang, Y.; and Liu, H. 2017. Attributed network embedding for learning in a dynamic environment. In *CIKM*. ACM.

Li, J.; Chen, C.; Tong, H.; and Liu, H. 2018. Multi-layered network embedding. In *SDM*. SIAM.

Linsker, R. 1988. Self-organization in a perceptual network. *Computer*.

Liu, W.; Chen, P.-Y.; Yeung, S.; Suzumura, T.; and Chen, L. 2017. Principled multilayer network embedding. In *ICDMW*. IEEE.

Ma, Y.; Ren, Z.; Jiang, Z.; Tang, J.; and Yin, D. 2018. Multi-dimensional network embedding with hierarchical structure. In *WSDM*. ACM.

Ma, Y.; Wang, S.; Aggarwal, C. C.; Yin, D.; and Tang, J. 2019. Multi-dimensional graph convolutional networks. In *SDM*. SIAM.

Meng, Z.; Liang, S.; Bao, H.; and Zhang, X. 2019. Co-embedding attributed networks. In *WSDM*. ACM.

Ni, J.; Chang, S.; Liu, X.; Cheng, W.; Chen, H.; Xu, D.; and Zhang, X. 2018. Co-regularized deep multi-network embedding. In *WWW*.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD*. ACM.

Qu, M.; Tang, J.; Shang, J.; Ren, X.; Zhang, M.; and Han, J. 2017. An attention-based collaboration framework for multi-view network representation learning. In *CIKM*. ACM.

Ravanelli, M., and Bengio, Y. 2018. Learning speaker representations with mutual information. *arXiv preprint arXiv:1812.00271*.

Ribeiro, L. F.; Saverese, P. H.; and Figueiredo, D. R. 2017. struc2vec: Learning node representations from structural identity. In *KDD*. ACM.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *ESWC*. Springer.

Shi, Y.; Han, F.; He, X.; He, X.; Yang, C.; Luo, J.; and Han, J. 2018. mvn2vec: Preservation and collaboration in multi-view network embedding. *arXiv preprint arXiv:1801.06597*.

Sun, Y.; Han, J.; Yan, X.; Yu, P. S.; and Wu, T. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB*.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *WWW*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *ICLR*.

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep graph infomax. *ICLR*.

Vinyals, O.; Bengio, S.; and Kudlur, M. 2015. Order matters: Sequence to sequence for sets. *NIPS*.

Wang, X.; Cui, P.; Wang, J.; Pei, J.; Zhu, W.; and Yang, S. 2017. Community preserving network embedding. In *AAAI*.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous graph attention network. In *WWW*. ACM.

Wang, D.; Cui, P.; and Zhu, W. 2016. Structural deep network embedding. In *KDD*. ACM.

Xu, L.; Wei, X.; Cao, J.; and Philip, S. Y. 2017. Multi-task network embedding. In *DSAA*. IEEE.

Yadav, P.; Nimishakavi, M.; Yadati, N.; Vashishth, S.; Rajkumar, A.; and Talukdar, P. 2019. Lovasz convolutional networks. In *AISTATS*.

Yang, C.; Liu, Z.; Zhao, D.; Sun, M.; and Chang, E. 2015. Network representation learning with rich text information. In *IJCAI*.

Zhang, H.; Qiu, L.; Yi, L.; and Song, Y. 2018a. Scalable multiplex network embedding. In *AAAI*.

Zhang, Z.; Yang, H.; Bu, J.; Zhou, S.; Yu, P.; Zhang, J.; Ester, M.; and Wang, C. 2018b. Anrl: Attributed network representation learning via deep neural networks. In *IJCAI*.

Zhou, S.; Yang, H.; Wang, X.; Bu, J.; Ester, M.; Yu, P.; Zhang, J.; and Wang, C. 2018. Prre: Personalized relation ranking embedding for attributed networks. In *CIKM*. ACM.

Zhuang, C., and Ma, Q. 2018. Dual graph convolutional networks for graph-based semi-supervised classification. In *WWW*.