A Novel Framework for Node/Edge Attributed Graph Embedding

Guolei Sun(✉) and Xiangliang Zhang

King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
{guolei.sun,xiangliang.zhang}@kaust.edu.sa

Abstract. Graph embedding has attracted increasing attention due to its critical application in social network analysis. Most existing algorithms for graph embedding utilize only the topology information, while recently several methods are proposed to consider *node content information*. However, the copious information on *edges* has not been explored. In this paper, we study the problem of representation learning in *node/edge attributed graph*, which differs from normal attributed graph in that *edges* can also be contented with attributes. We propose **GERI**, which learns graph embedding with rich information in node/edge attributed graph through constructing a heterogeneous graph. GERI includes three steps: construct a heterogeneous graph, take a novel and biased random walk to explore the constructed heterogeneous graph and finally use modified heterogeneous skip-gram to learn embedding. Furthermore, we upgrade GERI to semi-supervised GERI (named SGERI) by incorporating label information on nodes. The effectiveness of our methods is demonstrated by extensive comparison experiments with strong baselines on various datasets.

Keywords: Graph embedding · Node/edge attributed graphs · Network analysis

1 Introduction

Graph embedding, aiming to learn low-dimensional representations for nodes in graphs, has attracted a lot of attention recently due to its success in network learning tasks such as node classification [14], and link prediction [10]. Inspired by natural language models [9], Deepwalk is proposed to learn node embedding from network topology [11]. Then LINE [16] proposed to learn embedding by encoding first-order proximity and second-order proximity between nodes. Node2vec [2] improved Deepwalk [11] by introducing a more flexible random walk.

There is a new trend to integrate multiple types of input information including network topology and node content [3,20], neighbors homophily [22] or node labels [7,19,21]. In reality, networks are complex in terms that not only *nodes* but also *edges* contain rich information. For example, in a coauthor network, the nodes representing authors can be associated with a feature vector, which

Q. Yang et al. (Eds.): PAKDD 2019, LNAI 11441, pp. 169–182, 2019.
https://doi.org/10.1007/978-3-030-16142-2_14

contains information like affiliations or education background or research interest. Also, the edges indicating co-author relationships can be contented by the jointly published papers, which include key-words like classification, matrix completion etc. It is essential that graph embedding should learn from both *topology information* and *node/edge content information*.

There are several previous works considering attributed network embedding, where generally attributed network [5,12] is the network only with *node content information*. PPNE [6] uses node content information by enforcing representations to preserve the similarities between nodes. LANE [3] learns embedding by modelling node proximity in both attributed network space and label space. PLANETOID [21] uses deep neural networks to do semi-supervised representation learning, which utilizes text information as well as label information, and it considers multi-class classification problem. Generally, existing approaches have a common limitation: they cannot incorporate the *edge content information* and only consider *node attributes*.

In this paper, we extend the problem of attributed network embedding to a more general case, named *node/edge* attributed graph embedding, where not only *node*, but also *edges* can contain rich information. We propose a general framework for **g**raph **e**mbedding with **r**ich **i**nformation (called GERI), which can learn scalable representations for nodes in networks with rich text information on *nodes/edges*. By incorporating label information during representation learning process, we extend GERI to semi-supervised GERI (named SGERI). GERI and its variant are composed of three steps. Firstly, a homogeneous graph with text information on nodes/edges is converted into a heterogeneous one. The main advantage of this conversion is that it naturally integrates graph topology with node/edge content information or label information (only for SGERI), giving us an opportunity to exploit all such information and enhance the performance of learned representations. Then, a novel discriminant and flexible random-walk method is proposed to preserve the high-order similarity between nodes targeted for embedding, by exploring the constructed network in a mixture of the breadth first search (BFS) and the depth first search (DFS) manner. Finally, modified heterogeneous skip-gram model is used to learn the embedding for the nodes in the original network.

The evaluation of obtained graph embedding is conducted with multi-label/multi-class classification task on three datasets with *nodes/edges* information. The results show that GERI consistently and significantly outperforms state-of-the-art algorithms for various dimensions on all datasets in the unsupervised setting. SGERI in semi-supervised setting has the better performance than the semi-supervised methods including LANE and PTE. What's more, GERI, and SGERI are also computationally efficient since its major sections can be easily parallelized.

2 Related Work

The study in *unsupervised* representation learning with *only the topology* information has a big family of developed approaches [13]. The network topology is

usually represented by an adjacency matrix, $A_{(|V|*|V|)}$. To obtain node representation in $\mathbb{R}^d$, dimensionality reduction techniques like singular value decomposition or principal component analysis can be applied on graph Laplacian matrix and Modularity matrix [18]. However, the poor scalability and efficacy of these approaches makes them difficult to be applied to large-scale networks. Recently, another stream of work addresses the unsupervised representation learning of nodes in large-scale graph with an inspiration from neural language processing. Deepwalk [11] and node2vec [2] exploit word2vec [8,9] to learn embedding from word-context pairs sampled by random walks in the graph. And LINE [16] is proposed to explicitly preserve the first-order and second-order proximity between nodes.

The methods for *semi-supervised* representation learning with *only the topology* are also developed in order to incorporate label information. MMDW [19] jointly optimizes the max-margin classifier and the embedding learning model formulated as matrix factorization. Similarly, DDRW [7] jointly learns a classifier and vertex representation by combining the loss of SVM and Skip-gram model.

Then we introduce works which can exploit both *network topology* and *node features information*. TADW [20] considers node content information by decomposing an approximated word-context matrix, with the help of node information matrix as side information. HSCA [22] also follows matrix decomposition model and proposes to enforce homophily between nodes. An obvious weakness of both methods is that they require matrix operation like SVD decomposition, which prohibits them from dealing with large scale graphs. PPNE [6] is another method which belongs to this category. It proposes to preserve property similarity between nodes by adding inequality constraints or numeric constraints. Other works in this topic are semi-supervised. Yang et al. propose Planetoid for learning the representation for each graph node to jointly predict the class label and the neighborhood context in the graph [21], but the model is only designed for multi-class classification problem. LANE [3] learns embedding by modelling node proximity in both attributed network space and label space.

However, all the above-mentioned approaches are not able to incorporate information on edges, which can be integrated by our proposed model. In the first step of our model, we construct a heterogeneous graph to integrate information in both node and edges seamlessly.

3 Problem Formulation

Formally, let $G = (V, E, \mathbf{T}_V, \mathbf{T}_E)$ denotes a network with rich content information for nodes and edges. More specifically, $V = \{v_1, v_2, \ldots, v_{|V|}\}$ is a set of nodes, and $E = \{e = (v_i, v_j) : v_i, v_j \in V\}$ is a set of edges linking two nodes. $\mathbf{T}_V$ is node content attributes, e.g., the word occurrence matrix for nodes, where each entry $\mathbf{T}_V(i,k)$ indicates the occurrence of word w_k associating with node v_i, and $\mathbf{T}_V(i,k) = 0$ for the absence of w_k in v_i's content. $\mathbf{T}_E$ is the edge content attributes, e.g., word occurrence matrix for edges, where each entry $\mathbf{T}_E(i,j,k)$ indicates the occurrence of word w_k on edge connecting node v_i and v_j, and

$\mathbf{T}_E(i,j,k) = 0$ for the absence. The purpose of our work is to learn a low-dimensional representation vector $v \in \mathbb{R}^d$ for each node $v \in V$, by considering the network topology and rich text information on nodes ($\mathbf{T}_V$) and edges ($\mathbf{T}_E$). Note that $\mathbf{T}_V$ and $\mathbf{T}_E$ can be constructed by any attributes, not just text words that are used for simplifying model explanation.

Definition 1: Node/edge attributed graph: It differs from normal attributed graph in that not only nodes, but also edges can be associated with attributes.

Definition 2: Target nodes, Bridge nodes and Label nodes: *Target* nodes are the nodes in V in the original homogeneous network G, for which embedding will be learned. When converting G into a heterogeneous one, *bridge* nodes are created to incorporate the text information on nodes/edges, for assisting the embedding learning of target nodes. An example is shown in Fig. 1. The details of bridge nodes construction will be introduced in Sect. 4.1.

4 Method

4.1 GERI

Heterogeneous Network Construction. Given an attributed network $G = (V, E, \mathbf{T}_V, \mathbf{T}_E)$, we construct a bipartite heterogeneous network (V, U, E_{he}), where V includes target nodes, U contains bridge nodes, E_{he} are edges between V and U. Bridge nodes are the set of words, $U = \{w_1, w_2, \ldots, w_{|U|}\}$, existing in node and edge text information.

An edge in E_{he} connects a target node v_i and a bridge node w_k under two circumstances: (1) v_i and w_k are connected when $\mathbf{T}_V(i,k) \neq 0$. That is to say, a target node v_i is connected with a bridge node w_k if word w_k occurs in the content information of node v_i. The weight associating with the edge is the value of $\mathbf{T}_V(i,k)$. (2) v_i and v_j are both connected to w_k, when $\mathbf{T}_E(i,j,k) \neq 0$. In other words, target node v_i and v_j are both connected with a bridge node w_k if word w_k occurs in the content information of the edge connecting v_i and v_j.

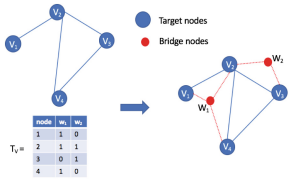

Fig. 1. Example of converting a homogeneous network (left) to a heterogeneous network (right) with bridge nodes.

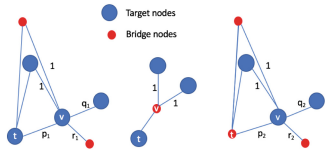

Fig. 2. Three cases of random walk in heterogeneous network, giving that random walk just reached v from t.

Then, the network our algorithm works on is $G_{he} = (V, U, E_{he}, E)$, which includes E_{he} and original E, and is associated with a mapping function $\varphi(v)$: $V\&U \rightarrow T = \{target, bridge\}$. A toy example is shown in Fig. 1. One prominent advantage of using the constructed heterogeneous network is that the text information is integrated seamlessly with the original network. There is no loss of information.

Modified Heterogeneous Skip-Gram. Given $G_{he} = (V, U, E_{he}, E)$, our goal is to learn a mapping: $v \rightarrow \mathbb{R}^d$ for target node $v \in V$. Besides, a bridge node can be also mapped to feature vectors in $\mathbb{R}^d$. We use $\mathbf{X}$ to represent the latent feature vector for $V\&U$ and $\mathbf{X} \in \mathbb{R}^{(|V|+|U|)*d}$. Inspired by metapath2vec [1], which formulated heterogeneous skip-gram and learn representation for nodes from meta-path. We maximizes the log-probability of observing network neighborhoods for all the nodes conditioned on their feature representation, and we formulate modified heterogeneous skip-gram in our constructed heterogeneous network as a maximum likelihood optimization problem with objective function defined as follows:

$$\arg\max_{\mathbf{X}} \sum_{v \in V} \sum_{t \in T} \sum_{n \in N_t(v)} \log(P(n|v; \mathbf{X})) + \lambda \sum_{v \in U} \sum_{t \in T} \sum_{n \in N_t(v)} \log(P(n|v; \mathbf{X})) \quad (1)$$

where $N_t(v)$ is the neighborhoods of node v, and has the type of t. As mentioned before, $t \in T = \{target, bridge\}$. The first and second part of the objective is the log-probability of observing network neighborhoods for target nodes and bridge nodes, respectively. λ is a balance parameter, controlling the weight of second part. It shows that λ does have a significant influence on the performance. We approximate $P(n|v; \mathbf{X})$ by negative sampling [9]. Then we use stochastic gradient ascent to get the $\mathbf{X}$. We formulate $\log(P(n|v; \mathbf{X}))$ as:

$$\log(P(n|v; \mathbf{X})) = \log(\sigma(\mathbf{X}_n \cdot \mathbf{X}_v)) + \sum_{m=1}^{M} \mathbb{E}_{u^m \sim P(u)}[\log(\sigma(-\mathbf{X}_{u^m} \cdot \mathbf{X}_v))] \quad (2)$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$, and $P(u)$ is the empirical unigram distribution defined on all nodes by viewing both target and bridge nodes homogeneously, where negative samples u^m will be drawn M times regardless of their types. Combining Eqs. (1) and (2), we can get the objective function for GERI.

An important component in the objectives of GERI is neighborhood $N_t(u)$, which has a significant influence on the embedding results. Inspired by node2vec [2], which proposed a concept of flexible neighborhood in homogeneous network, we propose a novel randomized procedure that can sample neighborhood of a source node in our constructed heterogeneous network.

Novel Sampling Strategy. Following but differing from node2vec, our proposed sampling method can explore the heterogeneous graph in a mixture of breadth first search (BFS) and depth first search (DFS), such that better neighbors of nodes can be obtained. Our sampling method is superior to the state-of-the-art sampling methods because the search method in node2vec [2] is designed

for homogeneous network and the existing sampling strategy in PTE [15] can only preserve the low proximity between nodes, which is usually not desirable.

Consider a random walk that just reached node v from node t in Fig. 2. Then it needs to decide where to go in the next step, which depends on the transition probability β_{vx} between node v and next node x, and the types of previously visited node v and t.

We define the transition probability β_{vx} in three cases:

Case 1: node t and v are both *target* nodes, as shown in the left example of Fig. 2. The next node to visit from v can be a target node, or a bridge node. We introduce three parameters p_1, q_1, and r_1 to guide the walk, and discuss their meanings later. Given the weight e_{vx} between node v and x, the transition probabilities β_{vx} is defined as:

$$\beta_{vx} = \begin{cases} p_1 * e_{vx} & \text{if } d_{tx} = 0 \\ 1 * e_{vx} & \text{if } d_{tx} = 1 \\ q_1 * e_{vx} & \text{if } d_{tx} = 2,\ x \in V \text{ target nodes} \\ r_1 * e_{vx} & \text{if } d_{tx} = 2,\ x \in U \text{ bridge nodes} \end{cases}$$

where d_{tx} denotes the shortest path distance between t and x.

Case 2: node t is a *target* node and v is a *bridge* node (the middle example in Fig. 2). In this case, we don't allow the walk to go back and expect the walk to explore more target nodes because we focus more on the relationship between a target node and other target nodes. β_{vx} is defined as:

$$\beta_{vx} = \begin{cases} 0 & \text{if } d_{tx} = 0 \\ 1 * e_{vx} & \text{if } d_{tx} \neq 0 \end{cases}$$

Case 3: node t is a *bridge* node and node v is a *target* node (the right example in Fig. 2). We introduce three parameters p_2, q_2, and r_2 to guide the walk. The transition probabilities β_{vx} is as follows:

$$\beta_{vx} = \begin{cases} p_2 * e_{vx} & \text{if } d_{tx} = 0 \\ 1 * e_{vx} & \text{if } d_{tx} = 1 \\ q_2 * e_{vx} & \text{if } d_{tx} = 2,\ x \in V \text{ target nodes} \\ r_2 * e_{vx} & \text{if } d_{tx} = 2,\ x \in U \text{ bridge nodes} \end{cases}$$

In the following, we discuss the meaning of the parameters and their implications.

Back parameter p. p_1 and p_2 control the probability to revisit the node that has been visited in the second last step. Setting it to a small value means that the walk is less likely to go back. However, setting it to a large value (>1) means the walk is more likely to visit the local neighbors of the source node. Then, it is more like the BFS search.

Out-target parameter q. q_1 and q_2, on the one hand, control the likelihood of visiting target nodes in the random walk. If $q_1(q_2)$ is greater than $r_1(r_2)$, then

Algorithm 1. GERI algorithm

Require: $G = (V, E, \mathbf{T}_V, \mathbf{T}_E)$, Dimensions d, walks per vertex γ, window size τ, walk length l, λ, and p, q,r
Ensure: matrix of nodes representation $\Theta \in \mathbb{R}^{|V|*d}$

```
Initialize Θ by standard normal distribution
Construct G_he = (V, U, E_he, E)
β=PreprocessBiasWeight(G_he, p, q, r)
for iter = 1 to γ do
    φ=shuffle(V)
    for all nodes v ∈ φ do
        walk=BiasedRandomWalk(G_he, β, v, l)
        trainpairs=GenerateSkipGramTraining(walk, τ)
        for (v1, v2) ∈ trainpairs do
            if v1 is a target node then
                SGD(k,d,(v1, v2),η)
            else
                SGD(k,d,(v1, v2),λη)
return Θ
```

the random walk is more likely to visit target nodes, which means target nodes play a more important role in the random walk. On the other hand, q_1 and q_2 control the depth of exploring the graph. If q_1 and q_2 are large, then the random walk is more likely to go as deep as possible, which is like DFS search.

Out-bridge parameter r. Contrary to q, r controls the likelihood of visiting bridge nodes in the random walk. If $q_1(q_2)$ is less than $r_1(r_2)$, then the random walk is more likely to visit target nodes. Similar to q, r controls the probability to explore the graph deeply. If it's high (>1), the walk is more like DFS. Otherwise, the walk is more like BFS.

In practice, since each pair of $p_1(p_2)$, $q_1(q_2)$ and $r_1(r_2)$ has the same meaning, we set $p_1 = p_2 = p$, $q_1 = q_2 = q$ and $r_1 = r_2 = r$.

GERI Algorithm and Complexity. We show the pseudo-code of GERI in Algorithm 1. It shows that GERI includes three steps: construct heterogeneous graph, conduct biased random walk, and then use modified heterogeneous skip-gram to learn embedding. The overall complexity of GERI is $O(|V|*\gamma*l^2)$, linear w.r.t. $|V|$.

4.2 SGERI

GERI can be easily extended to consider node label information, resulting a semi-supervised GERI (named SGERI), which works on $G'_{he} = (V, U, L, E_{he}, E'_{he}, E)$, where $L = (l_1, l_2, \ldots, l_k)$ represents the labels of nodes (training data) in V, k denotes the number of labels for V and E'_{he} represents the edges between V and L.

Similar to GERI, the complexity of SGERI is also linear with respect to V and is also easily parallelizable and can be executed asynchronously.

5 Experiments

5.1 Dataset

We employ three benchmark networks with text information on *nodes/edges*. The first two networks, which are publicly accessible, contain *node information*. The last network which contains *edge information* was extracted from the source in Aminer [17].

Cora [20] contains 2708 publications from 7 classes and 5429 links. Each publication is described by a binary 1433-dimension feature vector.

DBLP [4] contains 27199 authors and 66832 links, representing co-authorship. Each node has some labels out of 4 labels, representing research areas of the author. Each author is described by a 3000-dimension feature vector.

Aminer: we constructed a co-author network from the source in Aminer [17], containing 20105 authors and 48944 links. Each link corresponds to a co-authored paper. After processing paper abstracts by removing stop words and stemming, we have each edge is associated with an 897-dimension feature vector. The labels of nodes are research fields of the author.

5.2 Comparison Algorithm

The proposed methods are compared with several sate-of-the-art embedding algorithms, which can be divided into four groups. Firstly, to investigate the contribution of node/edge information, we compare GERI++ with Deepwalk [11], Line [16], and node2vec [2]. Secondly, we also include node feature information and naive combination of node2vec feature with node feature information as baselines. Thirdly, to evaluate the power of constructed heterogeneous graph, we feed constructed heterogeneous graph directly to Deepwalk, Line and node2vec. Fourth, we compare GERI and SGERI with PTE [15], and LANE [3], which are regarded as state-of-the-art algorithms in attributed network embedding. The detailed descriptions are listed as follows.

Deepwalk & LINE & Node2vec [11]**:** apply on the original homogeneous graph and set length of random walk as 150, # of walk as 10 and # of negative sampling as 5.

Naive Combination: combine node2vec embedding and text information.

Deepwalk(hete) & LINE(hete) & Node2vec(hete): feed the constructed heterogeneous graph to Deepwalk, LINE and node2vec.

TADW [20]**:** the embedding is learned from matrix decomposition.

PTE(unsupervised) & PTE [15]**:** For PTE (unsupervised), we construct two bipartite heterogeneous networks(target-target, target-bridge) and restrain it as an unsupervised method; For PTE, we construct three bipartite heterogeneous networks (target-target, target-bridge, target-label) and thus it remains as a semi-supervised method.

LANE(unsupervised) & LANE [3]**:** LANE(unsupervised) uses network and node content information, while LANE not only uses network and node

Table 1. Comparison of Micro-F1 and Macro-F1 score on Cora datasets for different dimensions

Algorithm	Micro-F1				Macro-F1			
	d = 16	d = 32	d = 64	d = 128	d = 16	d = 32	d = 64	d = 128
Deepwalk	0.7569	0.7757	0.8013	0.8151	0.7421	0.7645	0.7917	0.8041
Line	0.7323	0.7179	0.7090	0.7127	0.7142	0.7080	0.7048	0.7045
Node2vec	0.7762	0.7936	0.8096	0.8206	0.7651	0.7829	0.8000	0.81
Text only	0.7242	0.7399	0.7344	0.6957	0.6989	0.718	0.7097	0.6651
Naive combination	0.7864	0.8070	0.8198	0.8148	0.7629	0.7898	0.8033	0.8012
Deepwalk(hete)	0.7858	0.8065	0.7951	0.7962	0.7648	0.7867	0.7757	0.7790
Line(hete)	0.7928	0.8131	0.7903	0.7866	0.7928	0.7927	0.7663	0.7703
Node2vec(hete)	0.8172	0.8131	0.8064	0.7920	0.7957	0.7948	0.7836	0.7689
TADW	0.6732	0.7736	0.825	0.8279	0.5676	0.7400	0.808	0.8093
PTE(unsupervised)	0.7256	0.6959	0.7293	0.7275	0.6931	0.6669	0.7058	0.7048
LANE(unsupervised)	0.6948	0.7843	0.8266	0.8371	0.6098	0.7549	0.8136	0.8275
GERI	**0.8639**	**0.8698**	**0.8699**	**0.8655**	**0.8501**	**0.8604**	**0.8563**	**0.8526**

content (if available), but also uses label information of training data. We did extensive grid search on parameters. For α_1, we search from 0.1 to 1, with step 0.1, and for α_2, we search over [0.01 0.1 1.0]. And for LANE, we also search over δ_1, and δ_2.

GERI & SGERI: we set # of walk, length of walk, # of walk and # of negative sampling, to be the same as Deepwalk and Node2vec, for fair comparisons. The balance coefficient λ is 1 (default) and we use grid search to tune only on p, q, and r.

All the representation vectors are finally normalized such that their L2-norm as 1. We use logistic classification to evaluate all the embeddings.

5.3 Performance of GERI

We report the performance of different methods under various embedding dimensions on Cora, DBLP and Aminer in Tables 1, 2 and 3, respectively. We use 50% data as training and another 50% as testing. In Table 3, LANE(unsupervised) uses only network structure because it can't use edge content. And we don't show Text-only and Naive Combination, because they are not applicable in Aminer, which contains *edge content*.

First, GERI consistently outperforms all baselines for various dimensions on three datasets. For Cora, its performance improvement over PTE(unsupervised) is at least 19% for all dimensions. And it outperforms unsupervised LANE by 24%, 11%, 5.2% and 3.4% for dimension 16, 32, 64, 128, respectively. For DBLP, it is better than PTE(unsupervised) and largely improve LANE(unsupervised) by at least 9.5% over all dimensions. For Aminer, it outperforms PTE(unsupervised) by 6.0%, 5.9%, 5.8%, and 4.3% on $d = 16$, 32, 64, and 128, respectively.

Table 2. Comparison of Micro-F1 and Macro-F1 score on DBLP datasets for different dimensions

Algorithm	Micro-F1				Macro-F1			
	d = 16	d = 32	d = 64	d = 128	d = 16	d = 32	d = 64	d = 128
Deepwalk	0.5600	0.5769	0.5839	0.6027	0.4552	0.4896	0.5114	0.5386
Line	0.5220	0.4939	0.4895	0.5080	0.4193	0.3920	0.3946	0.4291
Node2vec	0.5760	0.5860	0.5952	0.6112	0.4858	0.5040	0.525	0.5466
Text only	0.6113	0.6472	0.6698	0.6894	0.6044	0.6333	0.6521	0.6721
Naive combination	0.7440	0.7476	0.7524	0.7511	0.718	0.7233	0.7284	0.7300
Deepwalk(hete)	0.7555	0.7582	0.7684	0.7771	0.7299	0.7319	0.7451	0.7556
Line(hete)	0.7669	0.7703	*0.7792*	*0.7853*	0.7442	0.7479	*0.7578*	*0.7648*
Node2vec(hete)	0.7553	0.7623	0.7716	0.7787	0.7294	0.7387	0.7495	0.7569
TADW	0.5023	0.6031	0.6657	0.7179	0.4925	0.5904	0.6497	0.697
PTE(unsupervised)	0.7575	0.7585	0.7698	0.7848	0.7383	0.7393	0.7509	0.7664
LANE(unsupervised)	0.1894	0.2462	0.6745	0.7246	0.1377	0.1800	0.6287	0.6790
GERI	**0.7725**	**0.7791**	**0.7891**	**0.7939**	**0.7488**	**0.7586**	**0.7687**	**0.7742**

Table 3. Comparison of Micro-F1 and Macro-F1 score on Aminer datasets for different dimensions

Algorithm	Micro-F1				Macro-F1			
	d = 16	d = 32	d = 64	d = 128	d = 16	d = 32	d = 64	d = 128
Deepwalk	0.4564	0.4643	0.5015	0.5089	0.3354	0.3632	0.4109	0.4373
Line	0.2890	0.2902	0.3839	0.4356	0.1922	0.1995	0.2734	0.3368
Node2vec	0.4759	0.4968	0.5111	0.5335	0.3537	0.3961	0.4181	0.4582
Deepwalk(hete)	0.6600	0.6625	0.6696	0.6729	0.5951	0.6014	0.6113	0.6191
Line(hete)	0.6564	0.6646	0.6687	0.677	0.5849	0.6024	0.6121	0.6227
PTE(unsupervised)	0.6419	0.6485	0.6551	0.6728	0.5665	0.5805	0.5974	0.6209
LANE(unsupervised)	0.2571	0.2940	0.3617	0.4631	0.1412	0.1684	0.2476	0.3756
GERI	**0.6801**	**0.6867**	**0.6932**	**0.7027**	**0.6159**	**0.6241**	**0.6330**	**0.6514**

Second, Deepwalk(hete), Line(hete) and Node2vec(hete) all have very competitive performance and are better than Deepwalk, Line and Node2vec that are applied to the original homogeneous graph. It thus verifies that our constructed heterogeneous graph effectively integrates the network topology and rich text information. But since they are all inferior to GERI, we get that our proposed biased sampling method is better than the sampling methods in these approaches.

Last, we find that TADW and LANE(unsupervised) (both methods use matrix optimization to learn embeddings) perform very poorly with low dimensions such as $d = 16$ and 32, but perform well when dimension of

embeddings increase to 64 or 128. However, PTE(unsupervised), Deepwalk, Line and Node2vec have consistent performance for all dimensions. For example, the performance of TADW and LANE(unsupervised) with low dimension of 16 or 32 is worse than other baselines in both Cora and DBLP data set. But these two methods perform well when d increases to 64 and 128.

5.4 Performance of SGERI

We compare SGERI with GERI, and other semi-supervised methods such as LANE and PTE on dataset DBLP and Aminer. For fair comparisons, we used the same set of training and testing data for all methods and did grid search over parameters.

We show the results on DBLP in Fig. 3. It shows that SGERI improved GERI by more than 4% in Micro-F1 and 5% in Macro-F1 score for all dimensions. From the comparisons between semi-supervised methods, we see that PTE outperforms LANE, and SGERI improved PTE by 14%, 12%, 4%, and 2% in Micro-F1 score for dimension of 16, 32, 64, and 128, respectively. For Macro-F1, SGERI improved PTE by 14%, 12%, 5%, and 2% for dimension of 16, 32, 64 and 128, respectively. Conclusively, we get that SGERI consistently outperforms PTE and LANE in all dimensions, and interestingly its superiority is more obvious in the setting of low dimension. The reason why SGERI is better than PTE is that it can better preserve proximity between nodes, which uses novel biased random walk and can take advantage of the high-order proximity while PTE only uses low-order proximity. For Aminer, the results are shown in Fig. 3. Similarly, the use of label information of training data really helps and largely improves the performance of our proposed methods. SGERI outperforms GERI by nearly 20% in both Micro-F1 and Macro-F1 score. Also, SGERI is better than PTE, with performance gain as least 7.1% for Micro-F1 and 9.8% for Macro-F1, for all dimensions. It further verifies that our sampling methods is better than the one in PTE.

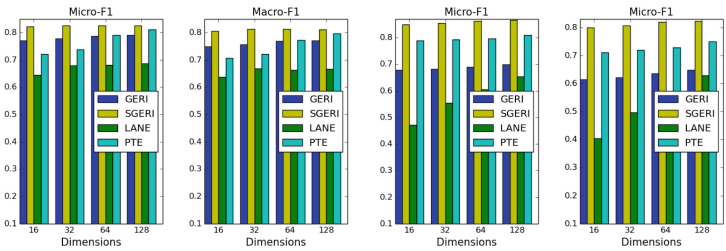

Fig. 3. Comparison between SGERI, GERI, LANE and PTE on DBLP dataset (Left two) and Aminer Dataset (Right two) over various dimensions

5.5 Parameter Analysis

We show effects of parameters in GERI. All experiments are done by setting d as 128.

Firstly, Fig. 4 shows that p, q, and r do have a significant influence on the performance. From the left plot in Fig. 4, we see that the setting of middle value for $1/p$ (from 1.0 to 2.0) and small value for $1/q$ (around 0.25) lead to better performance, which means a relatively high probability to explore target nodes when doing random walk better preserve the proximity between nodes. From the middle and right plots in Fig. 4, we find that a relatively small probability to explore the bridge nodes can give better performance. The underlying reason is that the information that bridge nodes contain is less important than target nodes. Considering the bridge nodes are the terms associated with nodes for this dataset, we can explain this from two aspects. First, the original homogeneous graph represents the co-authorship between authors, whose topology in an implicit way indicates the common research areas among authors. That's to say, the terms are supplementary for graph topology information even though they are the source of performance gain for our methods. Second, terms can be noisy. By limiting the probability of visiting bridge nodes, less noise will be brought to the embedding.

Next, we show how performance changes w.r.t. λ, walk length, # of walks and window size in Fig. 5. For λ, we see that good performance is obtained when lambda is a small value, i.e., 0.01 or 0.1. When λ further increases, F1 score drops dramatically. This is because λ controls the weight of loss function targeted on bridge nodes, and the information in bridge nodes is not as important as target nodes, following our discussion in above. For walk length, # of walks and window size, we see that the performance of node classification w.r.t. these three parameters follows similar pattern: performance increases sharply at the very beginning, increases slightly when we further increase parameter values, and fluctuates or converges or even decreases slightly in the later period.

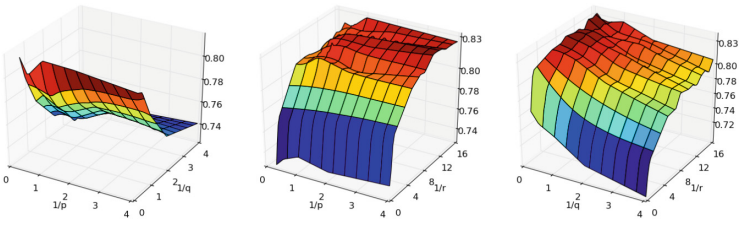

Fig. 4. Performance on different p, q, and r on Cora dataset: left (Micro-F1 score w.r.t. 1/p and 1/q); middle (Micro-F1 score w.r.t. 1/p and 1/r); right (Micro-F1 score w.r.t. 1/q and 1/r).

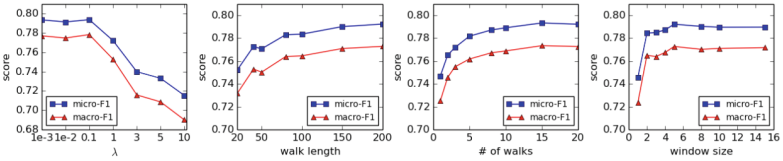

Fig. 5. Performance on different λ, walk length, the number of walks, and window size

6 Conclusion

We studied node/edge attributed graph embedding. GERI is proposed to firstly integrate original graph and copious information in *node/edges* into a heterogeneous graph, and then sample neighborhoods of nodes through the newly designed biased random walk. Finally, GERI learns embedding by modified heterogeneous skip-gram with negative samples. Furthermore, we develop SGERI which improves GERI by exploiting label information. For the future work, there are several possible directions. (1) consider dynamic nature of real graphs and the real-time changes of node/edge content information. (2) As is also the case with other attributed network embedding, we haven't considered the cases when node/edge content is not complete or contaminated.

Acknowledgement. This work is supported by King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. 2639.

References

1. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: scalable representation learning for heterogeneous networks. In: KDD, pp. 135–144 (2017)
2. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: KDD, pp. 855–864 (2016)
3. Huang, X., Li, J., Hu, X.: Label informed attributed network embedding. In: Proceedings of the Tenth International Conference on Web Search and Data Mining, pp. 731–739 (2017)
4. Ji, M., Sun, Y., Danilevsky, M., Han, J., Gao, J.: Graph regularized transductive classification on heterogeneous information networks. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010. LNCS, vol. 6321, pp. 570–586. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15880-3_42
5. Le, T.M., Lauw, H.W.: Probabilistic latent document network embedding. In: ICDM, pp. 270–279 (2014)
6. Li, C., et al.: PPNE: property preserving network embedding. In: Candan, S., Chen, L., Pedersen, T.B., Chang, L., Hua, W. (eds.) DASFAA 2017. LNCS, vol. 10177, pp. 163–179. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55753-3_11
7. Li, J., Zhu, J., Zhang, B.: Discriminative deep random walk for network classification. In: ACL, pp. 1004–1013 (2016)
8. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

9. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)
10. Pachev, B., Webb, B.: Fast link prediction for large networks using spectral embedding. arXiv preprint arXiv:1703.09693 (2017)
11. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: KDD, pp. 701–710 (2014)
12. Qi, G.J., Aggarwal, C., Tian, Q., Ji, H., Huang, T.: Exploring context and content links in social media: a latent space method. IEEE Trans. Pattern Anal. Mach. Intell. **34**, 850–862 (2012)
13. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**, 2323–2326 (2000)
14. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**, 93 (2008)
15. Tang, J., Qu, M., Mei, Q.: PTE: predictive text embedding through large-scale heterogeneous text networks. In: KDD, pp. 1165–1174 (2015)
16. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: WWW, pp. 1067–1077 (2015)
17. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: ArnetMiner: extraction and mining of academic social networks. In: KDD, pp. 990–998 (2008)
18. Tang, L., Liu, H.: Relational learning via latent social dimensions. In: KDD, pp. 817–826 (2009)
19. Tu, C., Zhang, W., Liu, Z., Sun, M.: Max-margin deepwalk: discriminative learning of network representation. In: IJCAI, pp. 3889–3895 (2016)
20. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.Y.: Network representation learning with rich text information. In: IJCAI, pp. 2111–2117 (2015)
21. Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings. In: ICML, pp. 40–48 (2016)
22. Zhang, D., Yin, J., Zhu, X., Zhang, C.: Homophily, structure, and content augmented network representation learning. In: ICDM, pp. 609–618 (2016)