Information Sciences 515 (2020) 341–351

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Relation constrained attributed network embedding

Yiqi Chen, Tieyun Qian*

School of Computer Science, Wuhan University, Hubei, China

ARTICLE INFO

Article history:
Received 24 May 2019
Revised 29 November 2019
Accepted 18 December 2019
Available online 19 December 2019

Keywords:
Attributed network embedding
Basic relation
Composite relation
Social network analysis

ABSTRACT

Network embedding aims at learning a low-dimensional dense vector for each node in the network. In recent years, it has attracted great research attention due to its wide applications. Most existing studies model the graph structure only and neglect the attribute information. Although several attributed network embedding methods take the node attribute into account, they mainly focus on the basic relations between the nodes and their attributes like a user and his/her interests (attributes). The composite relations between two nodes, and two nodes' attributes, and the related nodes and their attributes, contain rich information and can enhance the performance of many network analysis tasks. For example, two scholars having the common interests as "nature language processing" and "knowledge graph" may collaborate in the future and there will be a new edge in the network. However, such important information is still under-exploited.

To address this limitation, we propose a novel framework to exploit the abundant relation information to enhance attributed network embedding. The main idea is to employ the multiple types of relations in attributed networks as the constraints to improve the network representation. To this end, we first construct the composite relations between two nodes and their attributes in addition to the commonly used basic relations. We then develop a relation constrained attributed network (RCAN) framework to learn the node representations by constraining them with these relations. We conduct extensive experiments on three real-world datasets to show the effectiveness of our proposed RCAN as an attributed network embedding method for modeling various social networks. The results demonstrate that our method achieves significantly better performance than the state-of-the-art baselines in both the link prediction and node clustering tasks.

1. Introduction

Nowadays, information networks such as social networks, citation networks, protein networks, and user-item rating networks are becoming pervasive. The target of network embedding is to learn a low-dimensional dense vector for each node in the network. Network embedding is a fundamental problem in many network analysis tasks, and it has attracted great attention from researchers in the recent years [23,24,27,29,33].

Most network embedding methods focus on modeling the graph structure without considering side information like node attributes. More recently, several attributed network embedding (ANE) methods [20,21,35] have been shown to be effective in enhancing the performance of network analysis tasks by taking attribute information into account. However, these ANE

* Corresponding author.
E-mail addresses: yiqic16@whu.edu.cn (Y. Chen), qty@whu.edu.cn (T. Qian).

https://doi.org/10.1016/j.ins.2019.12.033
0020-0255/

methods only consider the *basic relations* like a user's neighbor node (friend) or his/her attribute (interest), and neglect the *composite relations* like the user's neighbor's attribute.

The composite relations convey abundant information beyond the basic relations. For example, if two scholars (two nodes in the network) sharing the common interests (these two nodes' attributes) as "nature language processing" and "knowledge graph", they are more likely to join the same community, or collaborate in the future. If a network embedding approach can encode such information into the node representations, it will definitely backbone the network analysis tasks such as clustering, community discovery, and link prediction.

To this end, we propose a novel framework to exploit various types of composite relations between two nodes, their attributes, and the related nodes and their attributes. Specifically, we first *construct composite relations* in attributed networks in addition to the basic relations. We then develop *a relation constrained attributed network embedding (RCAN)* method to encode the information contained in both the basic and composite relations. We finally successfully apply our proposed method to the link prediction, node clustering, and visualization tasks in attributed networks.

In summary, this paper includes the following contributions.

1. We propose an unsupervised attribute network embedding framework to exploit both the basic and composite relations in attributed networks.
2. We explore the basic and composite relations in attributed networks so as to preserve the abundant contextual information.
3. Extensive experiments on three real-world datasets demonstrate that our method is effective for various social network analysis tasks.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents our model in detail. Section 4 gives the experimental results. Finally Section 5 concludes the paper.

2. Related work

In this section we review the literature in network embedding. We first review the approaches in network embedding, and then move to the attributed network embedding.

2.1. Network embedding approaches

Network embedding has been applied to various tasks like link prediction [16], node classification [2], and community detection [22]. Traditional methods like locally linear embedding (LLE) [25] or Laplacian EigenMap [1] are developed based on dimensional reduction techniques.

Recently, a number of approaches have been proposed based on word2vec method [17], such as DW [23], LINE [27], SNBC [18], and node2vec [7]. After that, many deep learning based methods have been proposed for network embedding. Some of them enhance the representation by better exploiting the network structure. For example, SDNE [29] proposes a semi-supervised deep model to exploit the first-order and second-order proximity to preserve the network structure. SNBC [18] proposes a structural neighborhood-based classifier using a random walk for making decisions. HOPE [19] preserves asymmetric transitivity in directed graph by approximating high-order proximities. MNMF [31] preserves mesoscopic community structure instead of microscopic structure and proposes a modularity based community detection model. Struc2vec [24] proposes a novel and flexible framework to learn the representations that capture the structural identity of nodes in a network. HARP [3] proposes a multilevel graph representation learning paradigm by recursively coalescing the input graph into smaller but structurally similar graphs to capture the global structure of the input graph.

Another direction is toward the evolving networks. For example, NEU [33] proposes a network embedding update algorithm which implicitly approximates higher order proximities with theoretical approximation bound. It can be applied to any network representation learning methods for enhancing their performances. DynamicTriad [36] preserves both structural information and evolution patterns of a given network by imposing triadic closure process.

More recently, researchers propose to apply the newly developed deep learning technique like GAN to network embedding. For example, ANE [4] presents an adversarial network embedding framework which leverages the adversarial learning principle to regularize the representation learning. GraphGAN [30] proposes a graph representation model which unifies generative methods and discriminative methods to benefit from each other.

The above network embedding methods show significantly better performance than the traditional methods due to the combination of social or structural properties and deep neural networks. However, these methods focus on topological structure only and ignore attribute information. Hence they are inappropriate for modeling the networks with attribute information.

2.2. Attributed network embedding approaches

Attributed network embedding (ANE) takes both network structure and attribute information into account. The methods for ANE can be categorized into three types of methods, including supervised, semi-supervised, and unsupervised methods. Our RCAN model belongs to the unsupervised category.

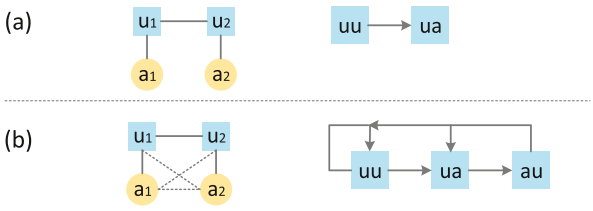

Fig. 1. Relations in an attributed network. Figure (a) denotes the basic (*uu, ua*) relations and Figure (b) denotes the extended network with a basic *au* relation and five composite relations, where the left and right part gives an instance and an abstraction of the corresponding relations, respectively.

The supervised and semi-supervised methods require label information to learn network embedding, and this limits its potential applications. Typical methods along this line include TriDNR [21], Planetoid-T [34], SEANO [14], and LANE [9]. TriDNR [21] adapts the skip-gram method [17] to combine the structure information, node content, and node label together. Planetoid-T [34] is a semi-supervised approach which leverages label information with node content and neighborhood context in the graph. SEANO [14] is also a semi-supervised method which exploits the property of outliers. LANE [9] is a label informed attributed network embedding method which jointly projects an attributed network and labels into a unified embedding space by extracting their correlations.

The unsupervised methods aim to learn embedding for an arbitrary network, and thus they have wide applications. For example, GAE [12] captures both topological and content information based on an autoencoder structure. VGAE [12] proposes a variational graph autoencoder to leverage structure and content information. SNE [15] learns the network representation by preserving both the structural proximity and attribute proximity. ARGA [20] is an adversarial graph embedding framework with graph autoencoder (ARGA) and variational graph autoencoder (ARVGA) as its two variants. DANE [5] is a deep neural network based method which captures the proximity in topological structure and node attributes. ANRL [35] uses a neighbor enhanced autoencoder to model the node attribute information based on an attribute-aware skip-gram method. Other directions along this line include accelerating [8,32] or exploring other information [10].

Despite the impressive progress in unsupervised ANE, the relations between nodes and their attributes have not been fully exploited. Existing methods focus on topological structure and just take attribute as the input or auxiliary component. Instead, we make full use of various types of composite relations between nodes and their attributes and hence our method can produce better representations for the nodes in the network.

3. Our proposed model

In this section we first introduce the composite relations and then present our model.

3.1. Basic and composite relation construction

Given an attributed network $\boldsymbol{G} = \{\boldsymbol{U}, \boldsymbol{UU}, \boldsymbol{A}, \boldsymbol{UA}\}$, where $\boldsymbol{U} = \{\boldsymbol{u_1}, \ldots, \boldsymbol{u_n}\}$ is the user set, $\boldsymbol{UU}$ is the (u_i, u_j) relation matrix, $\boldsymbol{A} = \{\boldsymbol{a_1}, \ldots, \boldsymbol{a_m}\}$ is the user's attribute set, and $\boldsymbol{UA}$ is the (u_i, a_j) relation matrix. For example, in a citation network, each node is a paper, and an edge is a citation between two papers, and the attribute is the keyword of the paper. In a social network, each node is a user, and an edge is a following relation between two users, and the attribute is the user's personal information.

For an attributed network $\boldsymbol{G}$, the basic objects are $u \in \boldsymbol{U}$ and $a \in \boldsymbol{A}$, and basic relations are $uu \in \boldsymbol{UU}$ (short for user-user relation) and $ua \in \boldsymbol{UA}$ (short for user-attribute relation). Most attributed network embedding methods [12,13,20] are built upon the above defined network $\boldsymbol{G}$, which rely on the basic relations as shown in the part (a) of Fig. 1. The *uu* relation is used to pass *ua*'s information in network without considering other basic relations like *au* (short for attribute-user relation) or composite relations like *uaau*.

On one hand, the basic *au* relation is another view from the attribute's point. For example, for a paper with a "knowledge graph" tag/attribute, it can be regarded as if there is a virtual edge between an attribute node "knowledge graph" and the paper, and all the papers with this attribute can be grouped together for further retrieval. On the other hand, there are abundant latent information hidden behind the composite relations. For example, two papers do not have direct link since they do not cite each other. However, with the same "knowledge graph" attributes, these two papers are semantically related. Clearly, if such composite relations can be further mined, the learnt embedding can preserve more information by putting such nodes in the neighboring positions in the latent space, Consequently, it can help improve the performance of social network analysis tasks.

Based on the above observations, we first let an attributed network $\boldsymbol{G}$ to contain one more basic relation matrix *AU* to represent the *au* relations. We then further extend $\boldsymbol{G}$ to include the composite relations, i.e., the five combinations of relations: (*uuua, uaau, uuuu, auua, auuu*), where *uuua* denotes the combinations of *uu* and *ua* relations. Both the basic *au* relation and five composite relations are shown in the part (b) of Fig. 1. We call such a extended graph as "*a relation graph*". For ease of presentation, we classify the composite relations into:

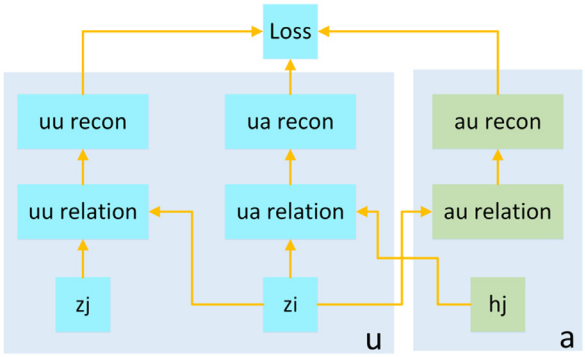

Fig. 2. The architecture of our RCAN-ba framework for the basic relations. The left component denotes the relations encoded from the point of view of a node (user) and the right component denotes the relations encoded from the point of view of an attribute.

user's composite relation:
(*uuua, uaau, uuuu*)
attribute's composite relation:
(*auua, auuu*)

These new relations contain more information than (*uu, ua*) since we can pass information through these new added relations. For example, instead of user's neighbors (*uu*) or user's attributes (*ua*), a user's new relation can explicitly represent a user's neighbors' neighbors (*uuuu*), or two users who share same attributes (*uaau*), or a user's neighbors' attributes (*uuua*).

3.2. RCAN framework: Learning from basic and composite relations

In this section, we present our relation constrained attributed network embedding (RCAN) framework for learning network representation from our proposed relations.

Constraining basic relations with RCAN-ba Firstly, we consider how to learn the network representation to reflect properties hidden in the basic relations. We call this method RCAN-ba and its framework is shown in Fig. 2.

Given an attributed network $\boldsymbol{G = \{U, UU, A, UA\}}$, we first assign embedding for each node (user) as $\boldsymbol{z_i} \in \mathbb{R}^{\boldsymbol{d}}$, where d is the latent dimension. Since we would encode the attribute information, we also assign embedding for each attribute as $\boldsymbol{h_i} \in \mathbb{R}^{\boldsymbol{d}}$. In order to constrain embeddings to follow the properties of basic relations, we define the probability of each relation based on the node and attribute embeddings. Take the relation *uu* as an example, for two nodes (i, j), we define the probability of the relation $R_{uu}(i, j)$ between them as Eq. (1):

$$P(R_{uu}(i, j)|\boldsymbol{z_i}, \boldsymbol{z_j}) = sigmoid(\boldsymbol{z_i z_j^T})),$$
$$sigmoid(x) = \frac{1}{1 + e^{(-x)}}, \tag{1}$$

where we use *sigmoid* to limit the value into the range of probability (0,1). Similarly, we can give definitions for other basic relations as:

$$P(R_{ua}(i, j)|\boldsymbol{z_i}, \boldsymbol{h_j}) = sigmoid(\boldsymbol{z_i h_j^T}),$$
$$P(R_{au}(i, j)|\boldsymbol{h_i}, \boldsymbol{z_j}) = sigmoid(\boldsymbol{h_i z_j^T}), \tag{2}$$

where $R_{ua}(i, j)$ denotes "the user's attribute" relation and $R_{au}(i, j)$ denotes "the attribute's user (who has this attribute)".

Next, we check whether these relations' probability represented by embeddings is accurate enough. To achieve this target, we use the real basic relations to constrain the probability. Take the *uu* relation as an example, we define the target function in Eq. (3):

$$max(p(\hat{\boldsymbol{UU}}|\boldsymbol{UU})) = max \prod_{i=1}^{n} \prod_{j=1}^{n} p(\hat{\boldsymbol{UU}}_{\boldsymbol{i,j}}|\boldsymbol{z_i}, \boldsymbol{z_j}),$$
$$p(\hat{\boldsymbol{UU}}_{\boldsymbol{i,j}} = 1|\boldsymbol{z_i}, \boldsymbol{z_j}) = P(R_{uu}(i, j)|\boldsymbol{z_i}, \boldsymbol{z_j}), \tag{3}$$

The intuition is that we try to maximize $p(\hat{\boldsymbol{UU}}|\boldsymbol{UU})$ to make our "represented probability of relation" close to the real relation. Similarly, we can also constrain the *ua* and *au* relations as follows:

$$max(p(\hat{\boldsymbol{UA}}|\boldsymbol{UA})) = max \prod_{i=1}^{n} \prod_{j=1}^{m} p(\hat{\boldsymbol{UA}}_{\boldsymbol{i,j}}|\boldsymbol{z_i}, \boldsymbol{h_j}),$$
$$p(\hat{\boldsymbol{UA}}_{\boldsymbol{i,j}} = 1|\boldsymbol{z_i}, \boldsymbol{z_j}) = P(R_{ua}(i, j)|\boldsymbol{z_i}, \boldsymbol{h_j}), \tag{4}$$

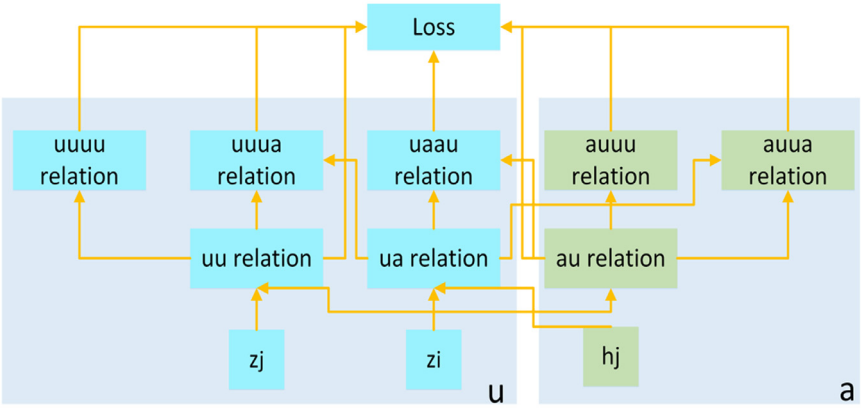

Fig. 3. The architecture of our RCAN-ca framework for the composite relations. The left component are encoding relations from the view of node(user) and right component are from the view of attribute.

$$max p(\hat{\boldsymbol{AU}}|\boldsymbol{AU}) = max \prod_{i=1}^{n} \prod_{j=1}^{m} p(\hat{\boldsymbol{AU}}_{\boldsymbol{i,j}}|\boldsymbol{h_i}, \boldsymbol{z_j}),$$
$$p(\hat{\boldsymbol{AU}}_{\boldsymbol{i,j}} = 1|\boldsymbol{h_i}, \boldsymbol{z_j}) = P(R_{au}(i, j)|\boldsymbol{h_i}, \boldsymbol{z_j}), \tag{5}$$

where $\boldsymbol{AU}$ is the transpose of $\boldsymbol{UA}$.

In the end, we design the loss function to be optimized in the neural network. Take the *uu* relation as an example, the loss function $\mathcal{L}^{uu}$ is defined as Eq. (6):

$$\mathcal{L}^{uu} = \sum\nolimits_{(i,j)\in\mathbb{R}^{n\times n}} \mathcal{L}^{uu}_{i,j},$$
$$\mathcal{L}^{uu}_{i,j} = -pw_{uu} * \boldsymbol{UU}_{\boldsymbol{i,j}} \log p(\hat{\boldsymbol{UU}}_{\boldsymbol{i,j}} = 1|\boldsymbol{z_i}, \boldsymbol{z_j})$$
$$-(1 - \boldsymbol{UU}_{\boldsymbol{i,j}}) \log(1 - p(\hat{\boldsymbol{UU}}_{\boldsymbol{i,j}} = 1|\boldsymbol{z_i}, \boldsymbol{z_j})), \tag{6}$$

where we apply the weighted binary cross-entropy loss to constrain the *uu* relation, and use pw_{uu} to control the weight of positive instances. pw_{uu} is defined as:

$$pw_{uu} = (n \times n - nz_{uu})/nz_{uu}, \tag{7}$$

where nz_{uu} is the number of non-zero instance in matrix $\boldsymbol{UU}$, pw_{uu} is designed to enhance the observed link (probability=1) and relax those unobserved link (probability=0). Similarly, we can get loss functions for other basic relations as $\mathcal{L}^{ua}$ and $\mathcal{L}^{au}$, thus the final loss function for basic relations is defined as:

$$\mathcal{L}_{basic} = \mathcal{L}^{uu} + \mathcal{L}^{ua} + \mathcal{L}^{au}, \tag{8}$$

where by minimizing $\mathcal{L}_{basic}$, we can constrain our embeddings to satisfy the properties of three kinds of basic relations.

Constraining composite relations with RCAN-ca With the composite relations, we can embed more abundant information than the basic relations, like the users sharing common neighbors (*uuuu*), and the attributes sharing common neighbors (*auua*). In this section, we further take the constructed composite relations into consideration. The proposed overall architecture, which we call RCAN-ca, is shown in Fig. 3.

Take the composite relation *uuuu* as an example. We define the probability for the relation of two nodes i and j which share a common neighbor k as follows:

$$g(\boldsymbol{z_i}, \boldsymbol{z_k}, \boldsymbol{z_j}) = \boldsymbol{z_i}\boldsymbol{z_k}^T\boldsymbol{z_k}\boldsymbol{z_j}^T, P(R_{uuuu}(i, j)|\boldsymbol{z_i}, \boldsymbol{z_j}) = sigmoid\left(\sum_{k\in U} g(\boldsymbol{z_i}, \boldsymbol{z_k}, \boldsymbol{z_j})\right) = sigmoid\left(\sum_{k\in U} \boldsymbol{z_i}\boldsymbol{z_k}^T\boldsymbol{z_k}\boldsymbol{z_j}^T\right), \tag{9}$$

where the function g is used to represent the probability between three node embeddings $(\boldsymbol{z_i}, \boldsymbol{z_k}, \boldsymbol{z_j})$. The rest composite relations can be defined in a similar way as that in Eq. (9).

We do not use the multiplication of two $P(R_{uu})_s$ in Eq. (1) to construct the $P(R_{uuuu})$ because this operation needs lots of computation resources. For the *uuuu* relation, the complexity of multiplication times of $(\boldsymbol{ZZ}^T)(\boldsymbol{ZZ}^T)$ is $\Theta(n^3 + 2dn^2)$ with Eq. (1). In contrast, with Eq. (9), we can change the calculation order as $(((\boldsymbol{ZZ}^T)\ \boldsymbol{Z})\boldsymbol{Z}^T)$ and the complexity will drop to $\Theta(3dn^2)$.

Next, we constrain the composite relations with the real links. Following the ideas of constraining basic relations, we try to maximize the target function of the *uuuu* relation as follows:

$$max(p(\boldsymbol{UU\hat{U}U}|\boldsymbol{UUUU})) = max\prod_{i=1}^{n}\prod_{j=1}^{n} p(\boldsymbol{UU\hat{U}U}_{\boldsymbol{i,j}}|\boldsymbol{z_i}, \boldsymbol{z_j}),$$
$$p(\boldsymbol{UU\hat{U}U}_{\boldsymbol{i,j}} = 1|\boldsymbol{z_i}, \boldsymbol{z_j}) = P(R_{uuuu}(i, j)|\boldsymbol{z_i}, \boldsymbol{z_j}), \tag{10}$$

where $\boldsymbol{UUUU}$ is the multiplication of two normalized $\boldsymbol{UU}$ matrix.

Similarly, we can model the attribute's relevant user's neighbors relation (*auuu*) as follows:

$$max(p(\boldsymbol{AU\hat{U}U}|\boldsymbol{AUUU})) = max\prod_{i=1}^{n}\prod_{j=1}^{n} p(\boldsymbol{AU\hat{U}U}_{\boldsymbol{i,j}}|\boldsymbol{h_i}, \boldsymbol{z_j}),$$
$$p(\boldsymbol{AU\hat{U}U}_{\boldsymbol{i,j}} = 1|\boldsymbol{h_i}, \boldsymbol{z_j}) = P(R_{auuu}(i, j)|\boldsymbol{h_i}, \boldsymbol{z_j}), \tag{11}$$

where the difference between Eqs. (10) and (11) lies in that Eq. (11) involves the user and attribute representation simultaneously while Eq. (10) only involves the user representation. The other three composite relations can also be modeled in a similar way as that in Eq. (11).

Then, we define the loss function using the above target function as:

$$\mathcal{L}^{uuuu} = \sum\nolimits_{(i,j)\in\mathbb{R}^{n\times n}} \mathcal{L}_{i,j}^{uuuu},$$
$$\mathcal{L}_{i,j}^{uuuu} = -pw_{uuuu}\boldsymbol{UUUU}_{\boldsymbol{i,j}} \log p(\boldsymbol{UU\hat{U}U}_{\boldsymbol{i,j}} = 1|\boldsymbol{z_i}, \boldsymbol{z_j})$$
$$-(1 - \boldsymbol{UUUU}_{\boldsymbol{i,j}}) \log(1 - p(\boldsymbol{UU\hat{U}U}_{\boldsymbol{i,j}} = 1|\boldsymbol{z_i}, \boldsymbol{z_j})), \tag{12}$$

where we use the similar definition as Eq. (6) to constrain the relation *uuuu*. The rest composite relations can be modeled following this procedure, and we can get the final loss function as follows:

$$\mathcal{L}_{all} = \mathcal{L}_{basic} + \mathcal{L}_{composite}$$
$$= (\mathcal{L}^{uu} + \mathcal{L}^{ua} + \mathcal{L}^{au}) + (\mathcal{L}^{uuuu} + \mathcal{L}^{uuua} + \mathcal{L}^{uaau} + \mathcal{L}^{auuu} + \mathcal{L}^{auua}), \tag{13}$$

where we combine the basic relations and composite relations together as the optimization target. The combination of RCAN-ba and RCAN-ca forms our RCAN model.

Training RCAN To train our RCAN model, we use the Adam optimizer [11] to update the parameters in the neural network. The training procedure is summarized in Algorithm 1.

Algorithm 1 RCAN Algorithm.

Require: attributed network $\boldsymbol{G} = \{\boldsymbol{U}, \boldsymbol{UU}, \boldsymbol{A}, \boldsymbol{UA}\}$
Ensure: embedding dimension d, node embedding matrix $\boldsymbol{Z}$, attribute embedding matrix $\boldsymbol{H}$
1: Initialize $\boldsymbol{Z}$, $\boldsymbol{H}$ with xavier [6] uniform distribution.
2: **repeat**
3: Forward-propagation, calculate the predicted probability of the basic and composite relations according to Eqs. (2) and (9)
4: Minimize $\mathcal{L}_{all}$ in Eq. (13), calculate the gradients
5: Back-propagation
6: Update the relevant parameters: $\boldsymbol{Z}$, $\boldsymbol{H}$
7: **until** Convergence
8: **return** $\boldsymbol{X}$

In line 1 in Algorithm 1, we initialize the parameters for our neural network using the xavier method [6]. This is a widely used strategy for training the neural network. It allows to keep the same variance of the weights gradient across layers during the training procedure and is better than the standard random initialization. From line 3 to 6, we train our model. We firstly calculate the predicted probability of the basic and composite relations in the forward propagation procedure. Then we calculate the relation constraint loss L_{all} in Eq. 13 to get the gradients. Finally, we back-propagate the gradients and update the relevant parameters. When the model converges, we return the node embedding matrix $\boldsymbol{X}$.

Since the calculation of neural network is affected by other factors like the optimization algorithm and tools, we focus on the times for matrix multiplication to calculate the algorithm complexity. The complexity analysis of our RCAN is shown as Eq. (14):

$$T(n, m, d) = \Theta(f_{basic} + f_{composite}) = \Theta(dn^2 + dmn + dmn)$$
$$+\Theta(3dn^2 + (2dn^2 + dmn) + (2dmn + dn^2) + 3dmn + (2dmn + dm^2))$$
$$= \Theta(7dn^2 + 10dmn + dm^2), \tag{14}$$

where f_{basic} and $f_{composite}$ represents the multiplication times from the basic relations (in Eq. (2)) and the composite relations (in Eq. (9)), respectively. d is the dimensionality for the latent embedding, n is the node number, and m is the attribute

Table 1
The statistics for datasets.

Dataset	Node	Edge	Content Words	Attributes
Cora	2708	5429	3,880,564	1433
Citeseer	3327	4732	12,274,336	3703
Pubmed	19,717	44,338	9,858,500	500

Table 2
Results for link prediction.

method	Cora		Citeseer		Pubmed	
	AUC	AP	AUC	AP	AUC	AP
LINE	0.8692	0.8989	0.8074	0.8583	0.8473	0.8801
DW	0.7923	0.8559	0.6660	0.7850	0.7616	0.8523
GAE	0.8970	0.9076	0.8772	0.8797	0.9620	0.9630
VGAE	0.9162	0.9295	0.8960	0.9075	0.9444	0.9462
ARGA	0.9135	0.9324	0.9143	0.9282	0.9485	0.9505
ARVGA	0.9209	0.9329	0.9144	0.9253	0.9050	0.9105
ANRL	0.8719	0.8664	0.9305	0.9306	0.9157	0.9093
RCAN-UU	0.7956	0.8491	0.7642	0.8206	0.7831	0.8439
RCAN-ba	0.8694	0.8861	0.9211	0.9335	0.9536	0.9586
RCAN	**0.9280**	**0.9435**	**0.9564**	**0.9641**	**0.9660**	**0.9709**

number. Since d is usually much smaller than n, i.e., $d \ll n$, the complexity is mainly determined by the larger one between n and m. If $m \ll n$, $T(n, m, d) = O(7dn^2)$; if $n \ll m$, $T(n, m, d) = O(dm^2)$. If we follow the calculation using Eq. (2), the complexity would be $\Theta(n^3 + (5d + 3m)n^2 + (8dm + m^2)n)$, which costs much more time.

4. Experimental evaluation

4.1. Datasets and baselines

Datasets We perform three typical analysis tasks, including link prediction, node clustering, and visualization, on three publicly available datasets. Their statistics are shown in Table 1. These datasets are attributed networks, with scientific publications as nodes, citation relationships as links, and unique words in each document as attributes [26].

Baselines For three social analysis tasks, we compare our method with the following state-of-the-art baseline methods.

DW [23]: an unsupervised network embedding method based on word2vec which focuses on modeling the network topological structure.

LINE [27]: an unsupervised network embedding method which takes the first and second order proximity of network structure into consideration.

GAE [12]: an unsupervised network embedding method based on an autoencoder framework, which considers both structure and content information.

VGAE [12]: an unsupervised network embedding method based on a variational graph autoencoder, which leverages structure and content information.

ARGA [20]: an unsupervised network embedding algorithm based on an adversarially regularized graph autoencoder, which takes both structure and attribute information into account.

ARVGA [20]: an variant of ARGA which adopts a variational graph autoencoder to learn the embedding.

ANRL [35]: an attributed network embedding method based on an attribute-aware skip-gram model to capture the network structure.

We do not compare with other network embedding methods like node2vec and SNE since they have been proven to be inferior to our baselines [20] [35]. Hence we only show the improvements over these baselines.

4.2. Link prediction

We report the AUC and average precision (AP) score and use the same dataset split and testing approach as those in [20]: 10% for testing, 5% for validation and the rest for training. We run 5 times with 5 random dataset splits to get the average score. For all the baselines, we use their recommended settings and get 32-dimensional node representation for link prediction task. For our method, we set learning rate=0.005, max iteration=200. For a fair comparison, we set the same dimension d=32 for all baselines and our method, which is the recommended setting in many baselines. RCAN-UU is the version that only the *uu* relation is used. RCAN-ba is the version that only uses the basic relations, and RCAN is the version that combines both the basic and composite relations.

Results for link prediction The results for link prediction are shown in Table 2. The best results are in bold.

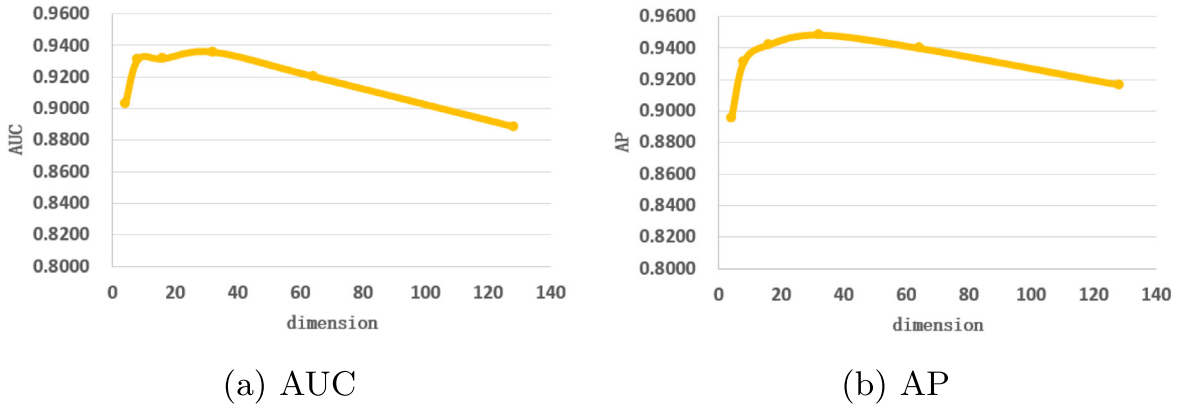

Fig. 4. Performance of link prediction with different embedding dimensions on Cora.

Table 3
Clustering results on Cora.

Cora	Acc	F1	Precision	NMI	ARI
LINE	0.5820	0.5762	0.6196	0.4129	0.3128
DW	0.6422	0.6317	0.6465	0.4287	0.3697
GAE	0.5414	0.5149	0.5645	0.3366	0.2312
VGAE	0.6034	0.5862	0.6001	0.4387	0.3764
ARGA	0.6928	0.6799	0.6829	0.4991	0.4521
ARVGA	0.6333	0.6324	0.6548	0.4663	0.3756
ANRL	0.5565	0.5695	0.6289	0.4128	0.3105
RCAN-UU	0.4346	0.4539	0.5137	0.2345	0.1445
RCAN-ba	0.6278	0.6139	0.6121	0.4062	0.3679
RCAN	**0.7637**	**0.7437**	**0.7757**	**0.5845**	**0.5734**

It is clear that by considering the composite relations between nodes and their attributes, our RCAN model achieves the best performance. RCAN significantly outperforms other baselines on Citeseer and Pubmed dataset (paired t-test at 0.05 level), and significantly outperforms other baselines except ARVGA and ARGA on Cora dataset.

Firstly, we compare the three versions of our RCAN. It is clearly that using the *uu* relation only is not enough and RCAN-UU performs poorly. However, when the attribute information is introduced into the basic relations, RCAN-ba shows great improvements over RCAN-UU. Finally, when the composite relations are employed, the performance of RCAN can be further enhanced on the basis of RCAN-ba and it beats all other baselines.

We then analyse the performance of the baselines. ARGA and GAE perform well on Cora and Pubmed datasets. The reason may be that they are both based on the basic gcn, which focuses mainly on the structure. However, on Citeseer dataset which has more attributes, ARGA and GAE are worse than ANRL which can better utilize attribute information.

In summary, our RCAN can consistently obtain the best results on different types of datasets by leveraging multiple basic and composite relations.

Parameter Study for Link Prediction We take the Cora dataset as an example to show the effects of the dimension of embedding and report the results in Fig 4.

We can find that even with only 4 dimension of embedding, our RCAN already performs pretty well. The overall results reveal the trends that: when adding the dimension of embedding from 4 to 32, the performance increases gradually, but the performance will deteriorate when we further increase the dimension. That is why we do not use more complex composite relations. This infers that the composite relations carry complementary information for attributed network embedding. However, the noises may be propagated with the composition of the basic relations.

4.3. Node clustering

We report results for node clustering in terms of five metrics: accuracy (Acc), precision, F-score (F1), normalized mutual information (NMI) and average rand index (ARI). We use the same dataset split and testing approach as those in [20]. For all the baselines, we use their recommended settings and get the 32-dimensional node embeddings for node clustering task. For our method, we use the same setting as that in link prediction. Since we find the results of node clustering fluctuate a lot for all the methods in different epochs, we report the best score of each method as the final result.

Results for node clustering The results for node clustering are shown in Tables 3–5. The best results are in bold.

Once again, our RCAN model is the best. While other baselines show good results on some datasets/metrics, only RCAN consistently perform well in terms of all metrics on all datasets. Compared to link prediction, the node clustering task is

Table 4
Clustering results on Citeseer.

Citeseer	Acc	F1	Precision	NMI	ARI
LINE	0.3475	0.3280	0.4539	0.1545	0.0640
DW	0.4142	0.3971	0.4304	0.1665	0.1298
GAE	0.3925	0.3754	0.3946	0.1883	0.1334
VGAE	0.5107	0.4915	0.5174	0.2482	0.2269
ARGA	0.5651	0.5512	0.5757	0.2911	0.2737
ARVGA	0.6228	0.5877	0.5979	0.3506	0.3551
ANRL	0.6126	**0.6007**	**0.6262**	0.3560	0.3370
RCAN-UU	0.2708	0.2707	0.3071	0.0491	0.0273
RCAN-ba	0.5326	0.5128	0.5310	0.2589	0.2368
RCAN	**0.6534**	0.5821	0.5831	**0.4183**	**0.4142**

Table 5
Clustering results on Pubmed.

Pubmed	Acc	F1	Precision	NMI	ARI
LINE	0.6242	0.6161	0.6446	0.2146	0.2075
DW	0.6476	0.6298	0.6622	0.2430	0.2593
GAE	0.6480	0.6538	0.6493	0.2402	0.2365
VGAE	0.6229	0.6251	0.6251	0.2214	0.2039
ARGA	0.6245	0.6242	0.6292	0.2144	0.2041
ARVGA	0.5964	0.5984	0.5985	0.1946	0.1765
ANRL	0.6628	0.6690	0.6664	0.2772	0.2652
RCAN-UU	0.4265	0.4065	0.5167	0.0545	0.0279
RCAN-ba	0.6369	0.6331	0.6704	0.3057	0.2556
RCAN	**0.6743**	**0.6718**	**0.7041**	**0.3419**	**0.2993**

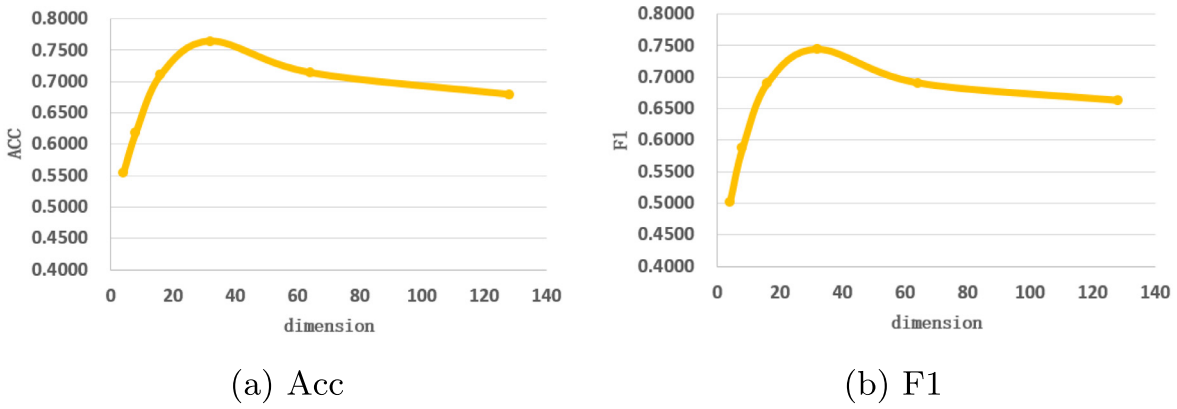

Fig. 5. Performance of node clustering with different embedding dimensions on Cora.

much harder since it can not learn task-related patterns during the unsupervised embedding learning process. That is also the reason for the fluctuation of results for all methods.

To learn the embedding for node clustering, a model which learns multi-type (structure and attribute) relevance between nodes will perform better than that mainly learns the single-type (structure or attribute) relevance. Indeed, the experimental results demonstrate the effectiveness of our framework by modeling the composite relations from both the structure and the attributes.

Parameter Study for Node Clustering We vary the dimension of embedding and report the results in Fig. 5.

The results reveal the similar trends as those in link prediction. When adding the dimension of embedding from 4 to 32, the performance increases gradually. However, the performance will deteriorate and become steady when we further increase the dimensionality. Basic and composite relations provide more rich information for node clustering task, but also bring in more noises.

4.4. Visualization

We report the visualization results for Cora dataset with t-sne algorithm [28]. The results are shown in Fig. 6.

It is clear that our RCAN model shows a more meaningful layout than other baselines. LINE and DW do not show useful layout since they do not utilize attribute information. GAE, VGAE, and ANRL have too many overlap parts. ARVGA's green

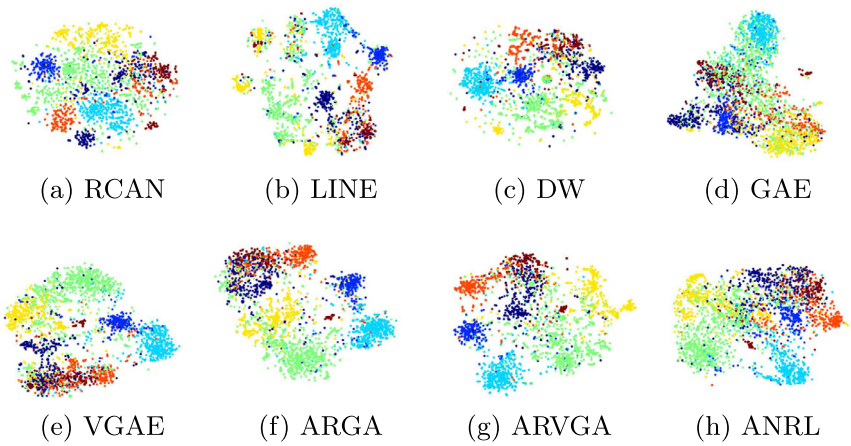

Fig. 6. Visualization result on Cora dataset. Each color denotes one group.

groups are separated. ARGA is the best among the baselines, but it has a large overlap in the upper left part. Our RCAN can show a good result since it fully mines both the basic and composite relations between nodes and attributes. Also, we constrain the embeddings directly instead of using a feature selection modle like gcn. Hence the learnt representations can well capture the properties of basic and composite relations.

5. Conclusion

In this paper, we propose a novel relation constrained attributed network embedding (RCAN) framework. In particular, we take the relations between users and attributes into account to make full use of node and attribute information. We construct both the basic and composite relations between nodes and attributes. We then propose a relation constrained method to exploit both the basic and composite relations. We combine all these relations' constraint together to get the final embedding. We conduct extensive experiments on three real-world networks. The experimental results demonstrate that our model significantly outperforms the state-of-the-art baselines.

In the future, we plan to investigate how to incorporate various types of relations more effectively. Different relations may play different roles in different data. How to make them complement each other in different situations is a challenging task. Moreover, while the basic and composite relations can be useful and the effective node representations can be learnt with these relation constraints, the composite relations might bring in noises if the attribute information is not that accurate. Hence the other direction is to discern the helpful relations and to discard the harmful ones.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Yiqi Chen: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing - original draft, Visualization. **Tieyun Qian:** Conceptualization, Methodology, Formal analysis, Writing - original draft, Writing - review & editing, Supervision, Project administration, Funding acquisition.

Acknowledgments

The work described in this paper is supported by the NSFC projects (61572376, 91646206), and the 111 project (B07037). The authors are very thankful to Prof. Huan Liu for his valuable comments and suggestions in improving the quality of the paper.

References

[1] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: Proceedings of the NIPS, 2001, pp. 585–591.
[2] S. Bhagat, G. Cormode, S. Muthukrishnan, Node classification in social networks, in: Proceedings of the SNDA, Springer, 2011, pp. 115–148.
[3] H. Chen, B. Perozzi, Y. Hu, S. Skiena, Harp: Hierarchical representation learning for networks, in: Proceedings of the AAAI, 2018.
[4] Q. Dai, Q. Li, J. Tang, D. Wang, Adversarial network embedding, in: Proceedings of the AAAI, 2018.
[5] H. Gao, H. Huang, Deep attributed network embedding., in: Proceedings of the IJCAI, 2018, pp. 3364–3370.

[6] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.
[7] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the SIGKDD, 2016, pp. 855–864.
[8] X. Huang, J. Li, X. Hu, Accelerated attributed network embedding, in: Proceedings of the SDM, SIAM, 2017, pp. 633–641.
[9] X. Huang, J. Li, X. Hu, Label informed attributed network embedding, in: Proceedings of the ICDM, ACM, 2017, pp. 731–739.
[10] X. Huang, Q. Song, J. Li, X. Hu, Exploring expert cognition for attributed network embedding, in: Proceedings of the ICDM, ACM, 2018, pp. 270–278.
[11] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
[12] T.N. Kipf, M. Welling, Variational graph auto-encoders, in: Proceedings of the NIPS, 2016.
[13] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of the ICLR, 2017, pp. 1–14.
[14] J. Liang, P. Jacobs, J. Sun, S. Parthasarathy, Semi-supervised embedding in attributed networks with outliers, in: Proceedings of the SDM, SIAM, 2018, pp. 153–161.
[15] L. Liao, X. He, H. Zhang, T.-S. Chua, Attributed social network embedding, arXiv preprint arXiv:1705.04969 (2017).
[16] D. Liben-Nowell, J. Kleinberg, The link-prediction problem for social networks, J. Assoc. Inf. Sci. Technol. 58 (7) (2007) 1019–1031.
[17] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the NIPS, 2013, pp. 3111–3119.
[18] S. Nandanwar, M.N. Murty, Structural neighborhood based classification of nodes in a network, in: Proceedings of the KDD, ACM, 2016, pp. 1085–1094.
[19] M. Ou, P. Cui, J. Pei, Z. Zhang, W. Zhu, Asymmetric transitivity preserving graph embedding, in: Proceedings of the KDD, ACM, 2016, pp. 1105–1114.
[20] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, C. Zhang, Adversarially regularized graph autoencoder for graph embedding, in: Proceedings of the IJCAI, AAAI Press, 2018, pp. 2609–2615.
[21] S. Pan, J. Wu, X. Zhu, C. Zhang, Y. Wang, Tri-party deep network representation, in: Proceedings of the IJCAI, AAAI Press, 2016, pp. 1895–1901.
[22] S. Papadopoulos, Y. Kompatsiaris, A. Vakali, P. Spyridonos, Community detection in social media, Data Min. Knowl. Discov. 24 (3) (2012) 515–554.
[23] B. Perozzi, R. Al-Rfou, S. Skiena., Deepwalk: Online learning of social representations., in: Proceedings of the SIGKDD, 2014, pp. 701–710.
[24] L.F. Ribeiro, P.H. Saverese, D.R. Figueiredo, struc2vec: Learning node representations from structural identity, in: Proceedings of the KDD, ACM, 2017, pp. 385–394.
[25] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000). 2323–6.
[26] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI Mag. 29 (3) (2008). 93–93.
[27] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the WWW, 2015, pp. 1067–1077.
[28] L. Van Der Maaten, Accelerating T-SNE using tree-based algorithms, J. Mach. Learn. Res. 15 (1) (2014) 3221–3245.
[29] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the KDD, ACM, 2016, pp. 1225–1234.
[30] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, M. Guo, Graphgan: graph representation learning with generative adversarial nets, in: Proceedings of the AAAI, 2018.
[31] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, S. Yang, Community preserving network embedding, in: Proceedings of the AAAI, 2017.
[32] W. Wu, B. Li, L. Chen, C. Zhang, Efficient attributed network embedding via recursive randomized hashing., in: Proceedings of the IJCAI, 2018, pp. 2861–2867.
[33] C. Yang, M. Sun, Z. Liu, C. Tu, Fast network embedding enhancement via high order proximity approximation., in: Proceedings of the IJCAI, 2017, pp. 3894–3900.
[34] Z. Yang, W. Cohen, R. Salakhudinov, Revisiting semi-supervised learning with graph embeddings, in: Proceedings of the ICML, 2016, pp. 40–48.
[35] Z. Zhang, H. Yang, J. Bu, S. Zhou, P. Yu, J. Zhang, M. Ester, C. Wang, ANRL: attributed network representation learning via deep neural networks., in: Proceedings of the IJCAI, 2018, pp. 3155–3161.
[36] L. Zhou, Y. Yang, X. Ren, F. Wu, Y. Zhuang, Dynamic network embedding by modeling triadic closure process, in: Proceedings of the AAAI, 2018.