Attributed Network Embedding for Incomplete Structure Information

Chengbin Hou[1,2] Shan He[2] Ke Tang[1]
[1]Southern University of Science and Technology [2]University of Birmingham
chengbin.hou10@foxmail.com s.he@cs.bham.ac.uk tangk3@sustc.edu.cn

Abstract

An attributed network enriches a pure network by encoding a part of widely accessible node auxiliary information into node attributes. Learning vector representation of each node a.k.a. Network Embedding (NE) for such an attributed network by considering both structure and attribute information has recently attracted considerable attention, since each node embedding is simply a unified low-dimension vector representation that makes downstream tasks e.g. link prediction more efficient and much easier to realize. Most of previous works have not considered the significant case of a network with incomplete structure information, which however, would often appear in our real-world scenarios e.g. the abnormal users in a social network who intentionally hide their friendships. And different networks obviously have different levels of incomplete structure information, which imposes more challenges to balance two sources of information. To tackle that, we propose a robust NE method called *Attributed Biased Random Walks (ABRW)* to employ attribute information for compensating incomplete structure information by using transition matrices. The experiments of link prediction and node classification tasks on real-world datasets confirmed the robustness and effectiveness of our method to the different levels of the incomplete structure information.

1. Introduction

Despite the success of Network Embedding (NE) for a pure network in the past five years, which aims to find a low-dimension vector representation of each node by considering *pure structure* information (Perozzi, Al-Rfou, and Skiena 2014; Tang et al. 2015; Grover and Leskovec 2016; Ou et al. 2016), NE for an *attributed network* by considering *both structure and attribute* information (Yang et al. 2015; Pan et al. 2016; Huang, Li, and Hu 2017; Gao and Huang 2018; Liao et al. 2018) has been recently attracted much more attention due to the following reasons.

From the input of NE perspective, an attributed network *enriches* a pure network by encoding a part of widely accessible node auxiliary information into node attributes, and previous works (Liao et al. 2018; Huang, Li, and Hu 2017) have shown the positive effect of additionally considering attribute information together with structure information. To define node attribute information: for a citation network, one may transform paper title into paper/node attributes (Pan et al. 2016); for a social network, one may encode user profiles into user/node attributes (Liao et al. 2018); and even for a pure network, one may encode node degree into node attributes (Kipf and Welling 2016).

From the output of NE perspective, the resulting node embeddings makes *downstream tasks* such as link prediction (Lü and Zhou 2011; Wei et al. 2017; Liao et al. 2018) and node classification (Huang, Li, and Hu 2017; Yang et al. 2015) more efficient and much easier to realize, since each node embedding is simply a *unified low-dimension vector representation*. In other words, the downstream tasks can be directly realized in Euclidean space by manipulating low-dimension data points that reflect both structure information and attribute information of an attributed network.

For simplicity, we may denote NE method by considering both structure and Attribute information as ANE method.

Incomplete Structure Information

In this work, we will investigate NE methods for an attributed network with incomplete structure information, more specifically, *without knowing complete structure/link* information. The similar incomplete attributed network has been discussed in context of community detection (Lin et al. 2012), but to our best knowledge, has not or rarely been discussed in context of NE. However, it is a worthwhile topic from the following two perspectives.

On the one hand, many real-world networks *inherently have incomplete* structure information. Just pick a few examples: 1) for a social network, some abnormal users such as criminals may intentionally hide their friendships; 2) for a terrorist-attack network where each node denotes an attack and two linked attacks are committed by the same organization, it is well-known that many attacks have not been clearly resolved (Lin et al. 2012); and 3) for a user friendship network extracted from an online website, there are some isolated users who have just registered or have never added friends due to worrying about personal privacy.

On the other hand, it becomes *harder to crawl complete* network datasets due to the awareness of data protection and the development of anti-crawler technique, especially crawling high quality network datasets from word-leading companies such as Facebook and Tencent.

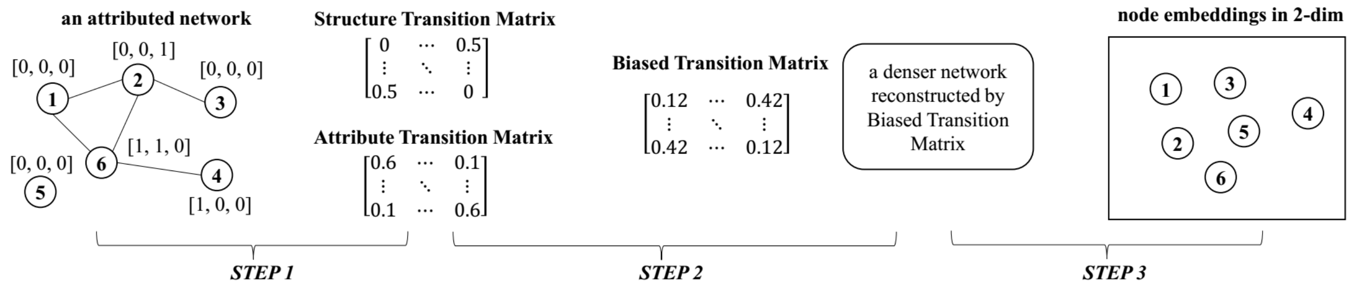

Figure 1: The illustration of ideas of our method: STEP-1. obtain the two transition matrices according to linkage and auxiliary information given by an attributed network respectively; STEP-2. calculate Biased Transition Matrix by weighted summing the corresponding rows of two transition matrices for each non-isolated node, and directly assigning the corresponding row of Attributed Transition Matrix for each isolated node; STEP-3. train node embeddings by applying a random-walks pure-structure based NE method to the denser network reconstructed by Biased Transition Matrix

More importantly, various real-world networks will lead to *different levels of incomplete* structure information, which imposes more challenges to ANE, as it becomes harder to balance two sources of information. Nevertheless, the essence of ANE is indeed to somehow properly combine structure and attribute information (Hamilton, Ying, and Leskovec 2017b; Cai, Zheng, and Chang 2018) so that the resulting node embeddings can help downstream tasks achieve superior performance.

In order to tackle the above challenge, the aim of this work is to design *a robust ANE method for attributed networks with different levels of incomplete structure information, so that it can deal with a wider range of different real-world networks with from relatively incomplete to highly incomplete structure information*[1].

Our Idea

To design such a robust ANE method, some desired properties might be as follows: 1) it apparently needs attribute information to help the incomplete structure information; 2) the use of attribute information should not depend on the use of incomplete structure information; 3) it should have a mechanism to deal with the isolated nodes which are often observed in real-world networks.

Our general idea is simple: employ attribute information to compensate incomplete structure information by using transition matrices. To be more specific, our method namely *Attributed Biased Random Walks (ABRW)* has three key steps as illustrated in Figure 1.

The contributions of this work are summarized as follows:

- We identify a significant problem of embedding attributed networks with incomplete structure information, which however, has been *largely ignored* in the past. And this is the *first work* dedicated to investigate the robustness of embedding methods.
- We propose a robust ANE method that applies attribute information to compensate incomplete structure information by *transition matrices*, which *offers a new thought* for combining multiple information. And the experiments confirmed the robustness and effectiveness of our method to the different levels of the incompleteness.
- We release the source code[2] of the proposed method on Github for benefiting the future research in NE field.

2. Related Works

In this section, we will review the related works from two perspectives: 1) the key ideas, and 2) why they are not ideal in case of incomplete structure information. Generally, they can be divided into four-folds.

Pure-structure based NE method: Although this category of methods ignores attribute information, they can still obtain node embeddings based on pure structure information. DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) applies truncated random walks to obtain node sequences and then, these node sequences a.k.a. network sentences are fed into the Word2Vec language model (Mikolov et al. 2013) so as to embed nodes closer if they co-occur more frequently on the sequences. Node2Vec (Grover and Leskovec 2016) can be viewed as an extension of DeepWalk due to that it employs more flexible truncated walks to explore and exploit network structure information and then, the resulting node sequences are fed into Word2Vec model for training node embeddings. Besides, there are some other successful pure NE methods such as LINE (Tang et al. 2015) and HOPE (Ou et al. 2016). Nevertheless, these methods are not robust to different levels of incompleteness, since they can only utilize the incomplete structure information.

Convex Optimization based ANE method: TADW (Yang et al. 2015) and AANE (Huang, Li, and Hu 2017) are two state-of-the-art methods in this category. In general, they first transform structure and attribute information into two matrices respectively, and then, formulize ANE problem as a convex optimization problem where the objective is to jointly minimize the differences of some distance measure

[1]In this work, we are interested in incomplete structure information, since missing links directly lead to the incomplete structure information, whereas we may use the a part of complete auxiliary information to define complete attribute information.

[2]https://github.com/houchengbin/ABRW

between structure information matrix and embedding matrix, and meanwhile, between attribute information matrix and embedding matrix. Because it is less intuitive to explain the effect of the different levels of incompleteness regarding these methods, we will investigate them via experiments.

Deep Learning based ANE method: One of the representative methods of this category is ASNE (Liao et al. 2018) and the core idea is to use carefully-designed stacked neural networks to learn a mapping where the input and output are two node embeddings for a pair of linked nodes respectively. In other words, one link gives one training data, and obviously, incomplete structure information gives less training data. Another representative method DANE (Gao and Huang 2018) has the similar problem of lacking training data in case of incomplete structure information. Therefore, the different levels of incompleteness will affect these methods, especially when the training data is not enough for a deep learning model a.k.a. an overfitting problem.

Graph Convolution based ANE method: Two representative methods of this category are GCN (Kipf and Welling 2016) and graphSAGE (Hamilton, Ying, and Leskovec 2017a). They first define node neighborhoods a.k.a. receptive field for every node based on structure information, and then, aggregate node neighborhoods' attribute information for further computation e.g. elementwise-mean. These methods are not robust, since the different levels of incompleteness change the definition of nodes neighborhoods and hence, their attributes to be aggregated.

3. Attributed Biased Random Walks (ABRW)

In this section, we will introduce our method in great details and also justify why our method succeeds.

Problem Definition

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W, A)$ be a given attributed network where a set of nodes/vertices $\mathcal{V} = \{v_1, \ldots, v_n\}$ and the total number of nodes $|\mathcal{V}| = n$; a set of links/edges $\mathcal{E} = \{e_{ij}\}$ and the total number of links $|\mathcal{E}| = e$; the weight associated to each link is a scalar $w_{ij} \in W$; the attribute associated to each node is a vector $A_i \in A$; the subscripts $i, j \in \{1, \ldots, n\}$. Note that the attributed network could be either directed or undirected and either weighted or unweighted.

Definition 1. *Structure Information Matrix* $W \in \mathcal{R}^{n\times n}$: The structure information refers to network link information, which will be encoded into matrix W. There are several popular choices to encode structure information (Goyal and Ferrara 2018; Ou et al. 2016) such as the first order proximity that describes node pairs immediate link information, the second order proximity that describes node pairs one-hop link information, and the third order proximity that describes node pairs two-hop link information. In this work, we simply use the first order proximity to define W a.k.a. the Adjacency Matrix of a network.

Definition 2. *Attribute Information Matrix* $A \in \mathcal{R}^{n\times m}$: The attribute information refers to network auxiliary/side information associated with each node, which will be encoded into matrix A where each row $A_i \in \mathcal{R}^m$ corresponds to the node attribute information for node v_i. In this work, we are not interested in how to preprocess the raw attributes such as users' profiles and papers' titles, but instead, assume A is ready-to-use. If necessary, one may follow techniques used in (Pan et al. 2016; Liao et al. 2018) to preprocess raw attributes for obtaining A.

The purpose of ANE method is then to find a mapping $Z = f(W, A)$ where $Z \in \mathcal{R}^{n\times d}$ is *Node Embedding Matrix* and each row vector $Z_i \in \mathcal{R}^d$ corresponds to the specific node embedding for node v_i.

Problem Formulation

Let us now interpret our method ABRW. It is the *first time* of using transition matrix for combining structure and attribute information in context of ANE.

A *transition matrix* a.k.a. Markov matrix or stochastic matrix is a square matrix where each entry is a nonnegative real number, and in our work, each row of transition matrix gives *discrete probability distribution* $\pi_i \in \mathcal{R}^n$ to indicate the probability of a walker to the next node from node v_i.

Definition 3. *Structure Transition Matrix* $T^W \in \mathcal{R}^{n\times n}$: The physical meaning of this matrix is that, according to structure information, the choice of the next node from current node v_i will be sampled based on the discrete probability distribution given by the row vector T^W_i i.e. the i^{th} row of T^W. To calculate T^W, we have:

$$T^W_{i,j} = f_{rowNorm}(W) = \frac{W_{i,j}}{\sum_{j\in n} W_{i,j}} \tag{1}$$

where $f_{rowNorm}$ is a function operating on each row of W such that each row becomes a probability distribution and each entry $T^W_{i,j}$ is calculated as shown in Eq. (1). Note that for an undirected network, both W and T^W are symmetric matrices; whereas for a directed network, they would be asymmetric matrices.

Definition 4. *Attribute Similarity Matrix* $S^A \in \mathcal{R}^{n\times n}$: This matrix stores the similarity measurements of attribute information between every pair of nodes on a network and hence, the dimension of this matrix is n-by-n. Recall that the given attribute information is $A \in \mathcal{R}^{n\times m}$ where each row $A_i \in \mathcal{R}^m$ corresponds to the node attribute information for node v_i. To calculate S^A, we have:

$$S^A_{i,j} = f_{cosSim}(A) = \frac{A_i A_j^T}{|A_i||A_j|} \tag{2}$$

where f_{cosSim} is a function to measure cosine similarity between every pair of rows of A and the resulting S^A is a symmetric matrix where each entry $S^A_{i,j} \in [-1, +1]$ and the diagonal entry $S^A_{i,i} = +1$. Of course, one may try other similarity measure to define S^A.

Definition 5. A *sparse operator* $\Theta(\cdot)$: The aim of this operator is to make Attribute Similarity Matrix S^A sparser. In practice, S^A is often a very dense matrix with few zeros, which will lead to a high computational cost in subsequent sampling stage. Our sparse operator is defined as:

$$S^A_{i,j} = \Theta(S^A) = \begin{cases} S^A_{i,j} & if\ S^A_{i,j} \in \{top\ k\ of\ S^A_i\} \\ 0 & otherwise \end{cases} \tag{3}$$

where the sparse operator $\Theta(\cdot)$ operates on each row of S^A and remains the largest k values but sets the rest to zeros, which means only the top-k most similar nodes for a node from attribute perspective will be remained. In fact, k here controls the sparsity/complexity of S^A.

Definition 6. *Attribute Transition Matrix $T^A \in \mathcal{R}^{n\times n}$*: The physical meaning is almost the same as Structure Transition Matrix T^W, except T^A is from attribute information perspective. To calculate T^A, we have:

$$T^A_{i,j} = f_{rowNorm}\left(S^A\right) = \frac{S^A_{i,j}}{\sum_{j\in n} S^A_{i,j}} \tag{4}$$

Definition 7. *Biased Transition Matrix $T \in \mathcal{R}^{n\times n}$*: This transition matrix aims to combine and balance two sources of information. With the pre-calculated T^W and T^A, it can be easily calculated as follows:

$$T_i = \begin{cases} T^A_i & if\ T^W_i\ is\ all\ zeros \\ \alpha T^W_i + (1-\alpha)T^A_i & otherwise \end{cases} \tag{5}$$

where T_i, T^W_i and T^A_i are the i^{th} rows of the corresponding transition matrices. In case of a node without any link i.e. *isolated node*, the row vector T^W_i will be all zeros and we will directly assign attribute information T^A_i to T_i for compensation. For other cases, we will apply a hyper-parameter α to balance two sources of information.

Algorithm Implementation

For better reproducibility and understanding, we summarize the implementation details in three pseudocodes.

```
Algorithm 1: Biased Transition Matrix Calculation
  Input: Structure Information Matrix $W$; Attribute
         Information Matrix $A$; top-k value $k$;
         hyper-parameter $\alpha$
  Output: Biased Transition Matrix $T$
  compute $T^W$, $S^A$, $\Theta(S^A)$, and $T^A$ sequentially
    according to Eq. (1), (2), (3) and (4)
  obtain $T$ according to Eq. (5)
```

For Algorithm 1, the aim is to combine two sources of information using transition matrices. With the help of above definitions and equations, it is straightforward to follow.

For Algorithm 2, the aim is to obtain a set of walks based on Biased Transition Matrix T given by Algorithm 1. The function $AliasSample$ is employed to reduce the time complexity (Grover and Leskovec 2016) of sampling next node according to current nodes probability distribution.

For Algorithm 3, the aim is to train node embeddings by Word2Vev language model (Mikolov et al. 2013) using a set of walks given by Algorithm 2. It is also straightforward to follow from implementation perspective, thanks to the well-developed the Python library Gensim and its scalable and robust function $Word2Vec$. In NE field, many previous works (Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016; Pan et al. 2016) have applied Word2Vec model for training node embeddings, and the reasons have been explained in great details therein.

```
Algorithm 2: Weighted Random Walks
  Input: Biased Transition Matrix $T$; walks per node $r$;
         walk length $l$
  Output: a set $walks$
  initialize a set $walks = \{\}$
  for all $iter \in r$:
    for all $v_i \in \mathcal{V}$:          # every node in the network
      initialize a list $walk = [v_i]$
      for all $walk\_iter \in l$:
        $curr = walk[-1]$  #take the last node in the list
        $prob\_dt = T_{curr}$
        $next = AliasSample(prob\_dt)$  #get next node
        append $next$ to $walk$
      append $walk$ to $walks$
```

```
Algorithm 3: Training Node Embeddings by Word2Vec
  Input: node embedding dimension $d$; a set $walks$;
         context size $c$; other required Word2Vec
         parameters $w2v\_param$
  Output: node embeddings $Z \in \mathcal{R}^{n\times d}$
  obtain $Z = Word2Vec(walks, d, c, w2v\_param)$
```

Why Our Method Succeeds

From analytical point of view, here are some potential reasons of why our method succeeds.

First of all, the benefits of using *transition matrix* are as follows: 1) A transition matrix can encode the similarity measurements of node pairs for network structure or attribute information as shown in Eq. (1-4); 2) A transition matrix can flexibly combine two sources of information as shown in Eq. (5); and 3) From a transition matrix to node embeddings is not a difficult task by applying Weighted Random Walks for generating node sequences and by using Word2Vec for training embeddings as shown in Algorithm 2 and 3. Note that this process can be easily adapted from recent successful pure-structure NE methods such as DeepWalk and Node2Vec. In general, the idea is to simulate weighted random walkers on the network reconstructed based on transition matrix and then, observe the frequency of the co-occurred nodes such that the more frequently a pair of nodes co-occurs, the more similar they should be.

Secondly, there are inherently some *isolated nodes* in real-world networks e.g. we can observe them from the datasets used in later experiments, and there might be more isolated nodes in case of incomplete structure information. Many previous methods (Perozzi, Al-Rfou, and Skiena 2014; Kipf and Welling 2016; Huang, Li, and Hu 2017; Liao et al. 2018) have no special treatment to handle them. Nevertheless, our method can automatically detect and deal with isolated nodes as shown in Eq. (5).

Thirdly, by varying the *hyper-parameter* α for balancing two sources of information as shown in Eq. (5), our method would be very similar to pure-structure NE methods e.g. DeepWalk if set $\alpha = 1.0$, but differently, in case of isolated node, our method also applies full attribute information for compensation and hence, our method could obtain superior performance than pure-structure NE methods.

Finally, thanks to the *sparse operator* $\Theta(\cdot)$ and its parameter k for remaining top-k similar nodes of a node from attribute perspective as shown in Eq. (3), we can easily control the sparsity of Attribute Similarity Matrix S^A, and hence, Attribute Transition Matrix T^A, and Biased Transition Matrix T. It is important for two main reasons: 1) Making S^A sparser not only remove the dissimilar nodes of a node, but also speed up the subsequent sampling process based on $\pi_i \in \mathcal{R}^n$ i.e. each row of T, as only a few more than k entries are nonzero out of totally n entries; and 2) Each row of T must not be an all-zero vector and therefore, the network reconstructed from T does not contain any isolated node, which is desirable for running a pure-structure NE method upon such a reconstructed pure network.

4. Experiments

In this section, we will use experiments to confirm the robustness and effectiveness of our method for the incomplete attributed network with incomplete structure information.

Experiment Settings

The datasets of attributed networks used in the experiments are summarized in table 1. These attributed networks have been widely tested in previous works (Yang et al. 2015; Huang, Li, and Hu 2017; Yang, Cohen, and Salakhudinov 2016; Kipf and Welling 2016; Velickovic et al. 2018).

	node	link	attribute	class	type
Cora	2708	5429	1433	7	directed
Citeseer	3327	4732	3703	6	directed
Pubmed	19717	44338	500	3	directed

Table 1: The summary of datasets used in the experiments

We compared our method ABRW with two baseline methods: DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) and AttrPure (ours) and also, with the state-of-the-art ANE methods: TADW (Yang et al. 2015), AANE (Huang, Li, and Hu 2017) and AttrComb (ours). We have also tried ASNE (Liao et al. 2018) and GCN (Kipf and Welling 2016), but they cannot obtain satisfactory results in case of insufficient links for the reasons as discussed in Related Works.

- DeepWalk uses pure structure information; while TADW, AANE, ABRW use both sources of information. Refer to Related Works for more details.
- AttrPure uses pure attribute information by applying SVD for dimension reduction to d-dim. We have also tried PCA for dimension reduction to d-dim, but they cannot dominate each other and hence, we only report SVD one.
- AttrComb uses both sources of information by applying DeepWalk to obtain $d/2$-dim embedding and AttrPure to obtain another $d/2$-dim embedding and then, concatenating them into one d-dim embedding. We have also tried elementwise-mean and elementwise-max, but concatenating way usually gives the better performance and hence, we only report the concatenating one.

For fairly comparison, we followed the suggestions of the original works to set parameters: 1) for all methods, the node embedding dimension $d = 128$; 2) for DeepWalk and ABRW, walks per node $r = 20$, walk length $l = 80$, and context size $c = 10$; 3) for TADW, AANE and ABRW, the hyper-parameters for balancing two sources of information are set to 0.2, 0.05 and 0.8 respectively. Note that all parameters were fixed throughout all the experiments.

We evaluated the resulting node embeddings via two downstream tasks: for *node classification* task i.e. a multiclass classification task, we take one-vs-rest Logistic Regression as the classifier and report micro-F1 or macro-F1 scores[3] (Grover and Leskovec 2016; Kipf and Welling 2016); for *link prediction* task i.e. a binary classification task, we employ cosine similarity as the measure and report AUC scores[4] (Liao et al. 2018; Yang et al. 2015). Note that all experiments were repeated for five times and we report their averages.

Data Preparation

In order to verify the *robustness* of the aforementioned methods, we need to simulate datasets so that they have different levels of incomplete structure information based on the real-world datasets as shown in Table 1.

During embedding phase, to simulate the *different levels of incomplete structure information*, we will *randomly remain different percentages of links* and only use the remained links as the input. Note that it will result in some isolated nodes, which is different from some previous works (Grover and Leskovec 2016; Liao et al. 2018) where they have to ensure one node at least keeps one link, since their methods cannot handle isolated nodes.

During downstream/evaluation phase: 1) For link prediction task, the dropped links will act as *positive samples*, and we also generate the *equal number* of non-existing links as *negative samples*, and finally, all samples will serve as ground truth; 2) For node classification task, we randomly remain a half of nodes' labels for training a classifier and then, the rest ones will serve as ground truth.

Experiment 1: Link Prediction (undirected)

In this experiment, similarly to most of previous works (Yang et al. 2015; Kipf and Welling 2016; Liao et al. 2018), we treat networks as undirected networks by adding another direction of link, if there is only one direction of link between a pair of nodes. Note that the undirected network only reflects whether two nodes has a relationship.

The results of link prediction task on the undirected networks are shown in Figure 2. Firstly, our method ABRW consistently outperformances other methods for all cases, and the performances are gradually improving as the available structure information increasing. Consequently, it validates the robustness and effectiveness of ABRW to the different levels of incomplete structure information.

[3]The key difference of two F1 scores is micro seeks the average among samples, whereas macro seeks the average among classes.

[4]AUC score is the area under the ROC curve that itself already considers various thresholds.

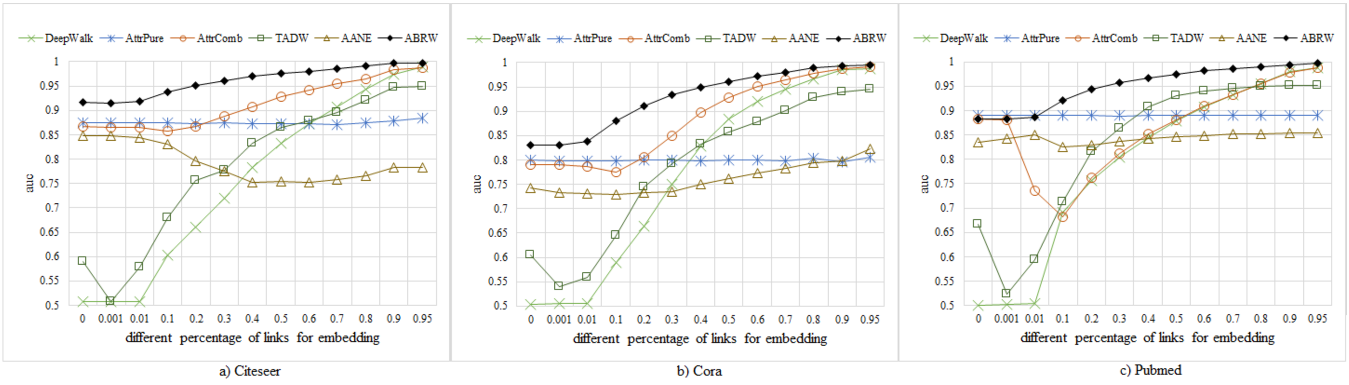

Figure 2: link prediction task on undirected networks by remaining different percentages of links for embedding

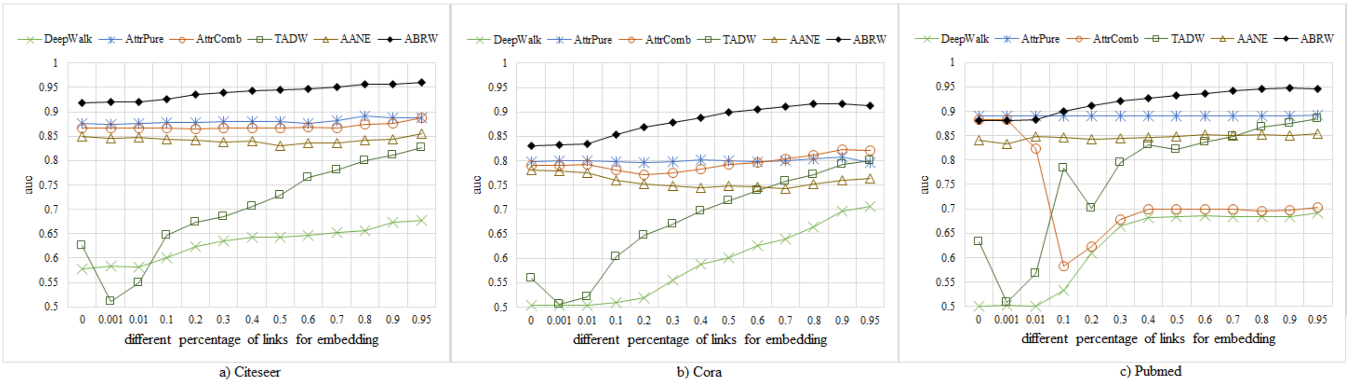

Figure 3: link prediction task on directed networks by remaining different percentages of links for embedding

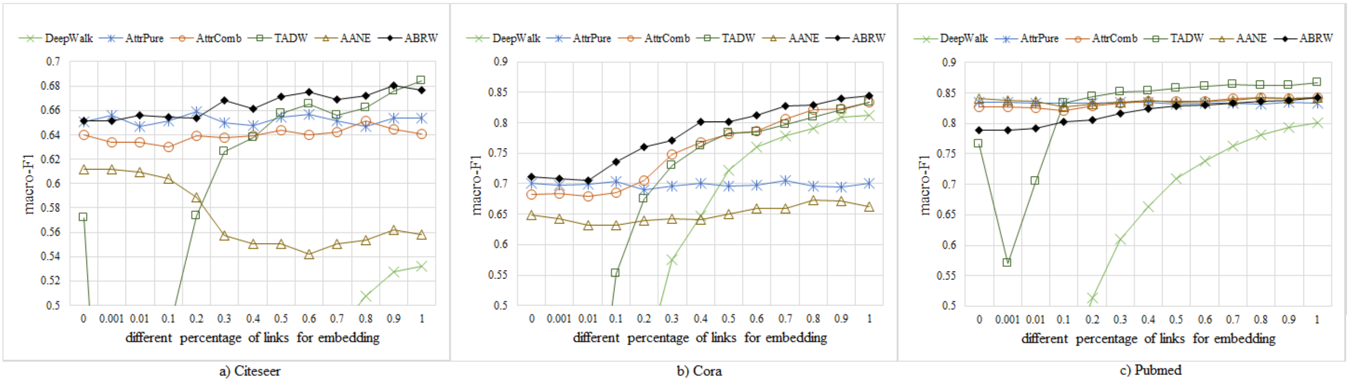

Figure 4: node classification task on undirected networks by remaining different percentages of links for embedding

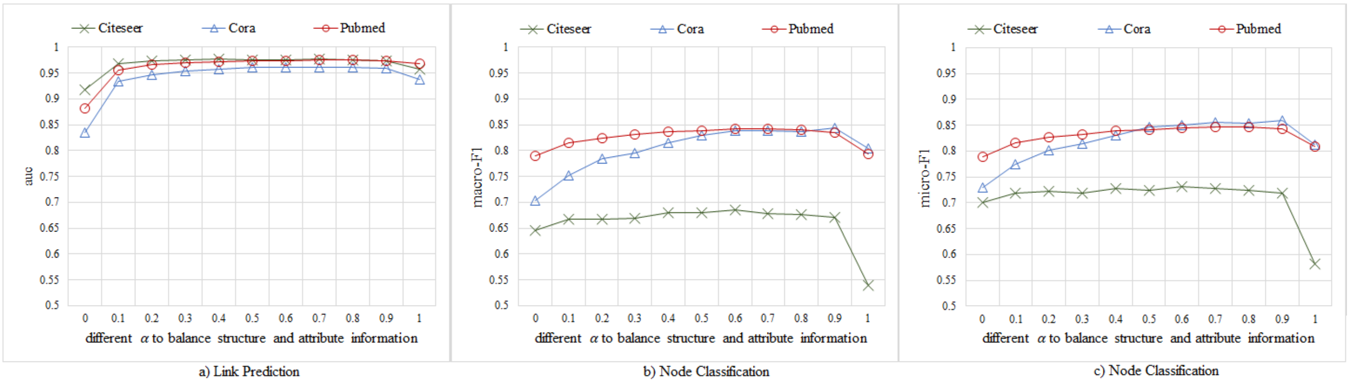

Figure 5: parameter sensitivity analysis of both link prediction and node classification tasks on undirected networks

Secondly, in cases of insufficient links, DeepWalk almost behaves like a randomly guessing method due to it only utilizes the incomplete structure information; whereas TADW can use two sources of information, but it still obtains poor performances, which implies TADW cannot properly combine two sources of information in such cases.

Thirdly, as the available structure information increasing, AANE and AttrComb have some strange behaviours that the performances may become worse, while TADW and ABRW always obtain better performances. Note that these four ANE methods all employ two sources of information, but they obtain quite different performances due to the different strategies used for combining information. Because ABRW achieves the best performances, it hence, indicates that our method of using transition matrices for combining information might be a more principled strategy.

Fourthly, the performances of AttrPure almost keep unchanged for all cases, but not surprisingly, the performances are not such good as others in case of sufficient links, since it ignores structure information. Therefore. it is useful of employing structure information to help attribute information during embedding.

Experiment 2: Link Prediction (directed)

In this experiment, differently from most previous works, we do not convert the given directed networks to undirected networks for three main reasons: 1) it is a more realistic scenario as the directed network also gives the direction; 2) it requires additional computational cost to covert a directed network to an undirected network and also, additional space cost to store the denser undirected network; 3) running embedding methods on a denser network increases running time especially for the methods using random walks.

Figure 3 shows the results of link prediction task on the directed networks. In general, the tendencies are very similar to Experiment 1, and this experiment also clearly validates the robustness and effectiveness of our method ABRW. Interestingly, AttrPure that only uses attribute information can beat most methods, which means other methods, except ABRW, cannot make use of the directed link/structure information to help attribute information.

Furthermore, we should mention that Experiment 2 is not to predict the directed link but instead, is to employ directed link information as one of inputs during embedding. Of course, predicting directed links might be a more useful task for directed networks, which will be left as future works and the key is to use an asymmetric similarity measure.

Experiment 3: Node Classification

We also evaluate the resulting node embeddings for node classification task as shown in Figure 4. In this experiment, 50% of node labels are used to train the Logistic Regression classifier. Note that we have also tried 30% and 70% of node labels for training Logistic Regression classifier, nevertheless, the tendencies are almost the same as 50% one and hence, we only report the 50% one.

For Citeseer and Cora datasets, ABRW outperformances other methods for almost all cases, which again validates the robustness and effectiveness of our method.

For Pubmed dataset, in case of sufficient links, TADW receives the best performances, which is at most 2% better than ABRW, AANE and AttrComb. However, in case of insufficient links, TADW receives very poor performances.

In general, regarding three datasets together, no method can always obtain the best performances for all cases. However, ABWR is worth to try in practice, since it obtains the best performances in many cases and for other cases, it can still obtain compatible performances.

Experiment 4: Parameter Sensitivity

In this experiment, we conduct both link prediction task (a half of links and the equal number of non-existing links serve as ground truth) and node classification task (a half of nodes' labels serve as ground truth) to analyze the sensitivity of hyper-parameter α used for balancing two sources of information. Differently from the above experiments, we vary α in X-axis and the results are shown in Figure 5.

Our method is robust to $\alpha \in \{0.1, 0.2, ..., 0.8, 0.9\}$ on both link prediction and node classification tasks, which means it can obtain satisfactory performance without too many tricks on parameter tuning. And this characteristic is desirable in recent AI community (Hutson 2018), since it makes the our method much easier to reproduce.

5. Conclusion and Future Works

In order to tackle the problem of embedding attributed networks with incomplete structure information, we proposed a robust ANE method called Attributed Biased Random Walks (ABRW) to apply attribute information for compensating incomplete structure information via transition matrices. The experiments of link prediction and node classification tasks confirmed the robustness and effectiveness of our method to the different levels of incomplete structure information. Besides, the experiments of parameter sensitivity analysis also verified the robustness of our method to a wide range of choices of the balancing hyper-parameter.

For future works, we noticed that our method as well as previous ANE methods employed the same coefficient to balance two sources of information for all nodes, however, we argue that the importance of attribute and structure information for each individual node in a network might be different e.g. a new user just registered in a website has richer attribute information, whereas a regular user may has richer structure information. One of interesting properties of using transition matrix is that: it is very easy to manipulate each row of transition matrix which corresponds to each individual node, and hence, we may use transition matrix to deal with the above problem. Furthermore, from the perspective of protecting personal privacy in the Internet, we may intentionally add/delete some links or generate noise on attributes to disturb embedding methods, and these actions can be leveraged on network transition matrix as well.

References

[Cai, Zheng, and Chang 2018] Cai, H.; Zheng, V. W.; and Chang, K. 2018. A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering*.

[Gao and Huang 2018] Gao, H., and Huang, H. 2018. Deep attributed network embedding. In *Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI))*.

[Goyal and Ferrara 2018] Goyal, P., and Ferrara, E. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151:78–94.

[Grover and Leskovec 2016] Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864. ACM.

[Hamilton, Ying, and Leskovec 2017a] Hamilton, W.; Ying, Z.; and Leskovec, J. 2017a. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 1024–1034.

[Hamilton, Ying, and Leskovec 2017b] Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017b. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

[Huang, Li, and Hu 2017] Huang, X.; Li, J.; and Hu, X. 2017. Accelerated attributed network embedding. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, 633–641. SIAM.

[Hutson 2018] Hutson, M. 2018. Artificial intelligence faces reproducibility crisis. *Science* 359(6377):725–726.

[Kipf and Welling 2016] Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[Liao et al. 2018] Liao, L.; He, X.; Zhang, H.; and Chua, T.-S. 2018. Attributed social network embedding. *IEEE Transactions on Knowledge and Data Engineering*.

[Lin et al. 2012] Lin, W.; Kong, X.; Yu, P. S.; Wu, Q.; Jia, Y.; and Li, C. 2012. Community detection in incomplete information networks. In *Proceedings of the 21st international conference on World Wide Web*, 341–350. ACM.

[Lü and Zhou 2011] Lü, L., and Zhou, T. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications* 390(6):1150–1170.

[Mikolov et al. 2013] Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

[Ou et al. 2016] Ou, M.; Cui, P.; Pei, J.; Zhang, Z.; and Zhu, W. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 1105–1114. ACM.

[Pan et al. 2016] Pan, S.; Wu, J.; Zhu, X.; Zhang, C.; and Wang, Y. 2016. Tri-party deep network representation. *Network* 11(9):12.

[Perozzi, Al-Rfou, and Skiena 2014] Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710. ACM.

[Tang et al. 2015] Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, 1067–1077. International World Wide Web Conferences Steering Committee.

[Velickovic et al. 2018] Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. In *International Conference on Learning Representations*.

[Wei et al. 2017] Wei, X.; Xu, L.; Cao, B.; and Yu, P. S. 2017. Cross view link prediction by learning noise-resilient representation consensus. In *Proceedings of the 26th International Conference on World Wide Web*, 1611–1619. International World Wide Web Conferences Steering Committee.

[Yang et al. 2015] Yang, C.; Liu, Z.; Zhao, D.; Sun, M.; and Chang, E. Y. 2015. Network representation learning with rich text information. In *Proceedings of the 24th International Conference on Artificial Intelligence*, 2111–2117. AAAI Press.

[Yang, Cohen, and Salakhudinov 2016] Yang, Z.; Cohen, W.; and Salakhudinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, 40–48.