Path Aggregation Model for Attributed Network Embedding with a Smart Random Walk

Xiang Xiao¹, Kang Zhang², Shuang Qiu* and Wei Liu¹

¹ School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, Guangzhou Province, 510000, China
² School of Systems Science and Engineering, Sun Yat-sen University, Guangzhou, Guangzhou Province, 510000, China
*Corresponding author’s e-mail: qiush9@mail.sysu.edu.cn

Abstract. Network embedding has attracted a surge of attention recently. In this field, how to preserve high-order proximity has long been a difficult task. Graph convolutional network (GCN) and random walk-based approaches can preserve high-order proximity to a certain extent. However, they partially concentrate on the aggregation process and sampling process respectively. Path aggregation methods combine the merits of GCN and random walk, and thus can preserve more high-order information and achieve better performance. However, path aggregation framework has not been applied in attributed network embedding yet. In this paper, we propose a path aggregation model for attributed network embedding, with two main contributions. First, we claim that there always exists implicit edge weight in networks, and design a tweaked random walk algorithm to sample paths accordingly. Second, we propose a path aggregation framework dealing with both nodes and attributes. Extensive experimental results show that our proposal outperforms the cutting-edge baselines on downstream tasks, such as node clustering, node classification, and link prediction.

1. Introduction
Network embedding represents nodes by low-dimensional vectors, and utilizes the obtained low-dimensional vectors for network analysis tasks, such as link prediction, node classification, community detection, and so on. In real world networks, the nodes are always attached with some extra information, e.g. texts, images, and other types of contents. Typically, the extra information is captured using attribute vectors associated with each node. Utilizing these attributes, network embedding can achieve better performances.

Many existing network embedding methods only consider the network structure. We call them the “plain network embedding” [1, 2, 3, 4, 5]. Recently, attributed network embedding has attracted more attention from the academic community, and an increasing number of models are proposed [6, 7, 8, 9, 10, 11, 12, 13, 14]. In the field of network embedding, a longstanding challenge is to preserve the high-order proximity. Early approaches based on matrix factorization [15, 16] consider the proximity with first-order neighbor nodes, and so do their variants for attributed networks [17]. Other approaches like LINE [2] and SDNE [4] take a step further, considering both first-order and second-order proximity. However, these methods ignore higher order proximity. Random walk-based methods [6, 7, 8] sample paths from the network, preserve the proximity of nodes in each path. GCN [9] and its descendants focus on aggregating the neighbor nodes recursively and obtains the high-order proximity after several epochs. Recently, the idea of path aggregation has attracted increasing attention [18, 19],
where path sampling and the aggregation process are combined to better capture high-order proximity. However, path aggregation framework has not been used in attributed network embedding yet.

In this paper, we propose a path aggregation model for attributed network embedding (PANE for short) with a tweaked random walk. Our contributions are threefold.

- We propose a tweaked random walk strategy to sample paths. We claim that there exists implicit edge weight in a network, and propose a new transfer probability for random walk, considering the impact of the implicit edge weight. We validate the effectiveness of the new random walk through ablation studies.
- We propose a path aggregation model for attributed network embedding, dealing with both high-order proximity and attributes information.
- We conduct extensive experiments against the state-of-the-art baselines on various tasks with real-world datasets. Our proposal achieves competitive performances.

2. Related Work

Plain Network Embedding. Network embedding can be traced back to matrix factorization methods [15, 16], which utilize eigen-decomposition to preserve the adjacency matrix. Then, LINE [2] applies two different objective functions to capture the first and second order proximity. SDNE [4] extends LINE by applying a deep autoencoder to preserve node proximity. DeepWalk [1] and Node2Vec [3] utilize random walk to sample global nodes and feed them to language models to get the embedding. DNE [20] proposes a degree-biased random walk, aiming to better capture the global structure. However, none of these random walk strategies takes the influence of dominant nodes into consideration.

Attributed Network Embedding. SNE [21] applies a multi-layer network to fuse the structure and attributes information. AANE [17] utilizes symmetric matrix factorization to represent the attribute similarity. CANE [7] extends LINE to learn local structural and textural information. However, these approaches can only capture the local information. TADW [6] utilizes text-associated random walk to incorporate text features. MMDM [22] applies a max-margin random walk to sample global nodes according to the label information. TriDNR [23] separately learns embeddings from the structure-based random walk and label-fused Doc2Vec attribute model. GCN-based methods use graph convolutional networks for network embedding [9], and learn the global information by aggregating the features of neighbor nodes. GraphSAGE [24] extends the GCN framework to the inductive setting. The variants GAE [11], VGAE [12], and AGE [10] comprise a GCN encoder and a reconstruction decoder.

3. Proposed Method

We first give the notations and define the problem, and then introduce our novel attributed network embedding model.
3.1. Notations

The attributed network can be represented as \( G = (V, E, A) \), where \( V = \{v_1, v_2, \ldots, v_n\} \) is the node set and \( n \) is the number of nodes. The set of neighbor nodes for \( v_i \) is denoted by \( N_i \). We denote by \( r_i \in \mathbb{R}^n \) the one-hot ID vector of node \( v_i \). \( \mathcal{E} = \{e_{i,j}\}_{i,j} \in \mathbb{R}^{n \times n} \) is the adjacency matrix and \( A = \{a_{i,j}\} \in \mathbb{R}^{n \times m} \) is the attribute matrix. If there exists an edge between \( v_i \) and \( v_j \), \( e_{i,j} > 0 \). Otherwise, \( e_{i,j} = 0 \). \( a_i \in \mathbb{R}^{m} \) is the attribute vector of node \( v_i \). The objective of network embedding is to cast each node \( v_i \) to a vector \( y_i \in \mathbb{R}^k \) (where \( k \ll n \)).

3.2. Overall Framework

We propose a path aggregation model for attributed networks, with an tweaked random walk algorithm to sample paths. As is illustrated in Figure 1, there are three modules in our proposal, (1) the local fusion module for fusing the local topology and attributes information of each node, (2) the random walk sampling module for sampling paths, i.e. the high-order neighbors, and (3) the path aggregation module for aggregating the high order neighbors and preserving high-order proximity.

3.3. A Tweaked Random Walk Revealing Implicit Edge-Weight

Random walk can be seen as a sampling process of the high-order neighbors. For traditional random walk strategy as proposed in DeepWalk [1], the next node is chosen randomly from the connected nodes. Then, some follow-up works improve the original random walk to sample nodes efficiently, i.e. LINE [2], node2vec [3], and VRRW [25]. The existing random walk strategies sample paths according to the topology of a network, which treats each edge equally. In the following, we claim that there exists implicit edge-weight in a plain network, and propose a tweaked random walk accordingly.

**Implicit Edge-Weight & Node Degree.** According to our observation, in real world networks, the influence of each node is vastly different. Some nodes are dominant and are more likely to affect other nodes. As a result, for one target node, the edge of a dominant node always shows larger influence than that of a non-dominant node. For instance, in a citation network, one literature (target node) cites many other literature (edges), but the influence (edge weight) of each reference is vastly different. The target literature is more likely to be influenced by the highly-cited literature in the references, and share its idea. The relation of two nodes is always much more complex than an unmarked edge, which provides only one bit information. We claim that there always exist implicit edge weights in a network, representing the different influence of one node to another node. In this paper, we take the underlying difference into consideration. In this paper, we consider the nodes with more neighbors as the dominant nodes. Namely, the implicit edge-weight is determined by the node degree \( d_i \).

**Degree-Based Random Walk Revealing Implicit Edge-Weight.** Given that the edge connecting with a node of a larger degree have a larger implicit weight, random walk strategy can be tweaked by increasing the transfer probability [25, 20] to nodes with larger degrees. Accordingly, we propose a novel random walk. For each pair of nodes \( (v_i, v_j) \) connected by an edge \( e_{i,j} \), we define a transfer weight \( M_{i,j} \) to represent the possibility of transferring from \( v_i \) to \( v_j \), which is calculated as follows,

\[
M_{i,j} = \begin{cases} 
    e_{i,j} \cdot \frac{d_j + \epsilon}{d_i + d_j} \cdot \theta, & \text{dis}(v_k, v_j) = 2 \\
    e_{i,j} \cdot \frac{d_j + \epsilon}{d_i + d_j}, & \text{dis}(v_k, v_j) < 2 
\end{cases}
\]  

As \( v_i \) and \( v_j \) represent the current node and one candidate of the next node respectively, \( v_k \) represents the last node prior to \( v_i \). \( d_i \) and \( d_j \) represent the degrees of \( v_i \) and \( v_j \), respectively. \( \text{dis}(v_k, v_j) \) denotes the shortest distance between node \( v_k \) and node \( v_j \). \( \epsilon \) is the smoothing parameter. \( \theta \) is the encouraging parameter. According to Equation (1), \( \theta \) encourages further walk steps, where \( v_k \) and \( v_j \) are not connected. The following briefly describes the two key points of the proposed random walk strategy:

**Impact of Implicit Edge Weight.** We increase the transfer probability according to the implicit edge weight. The equation \((d_j + \epsilon)/(d_i + d_j)\) is an increasing function of \( d_j \), which we regard as the
impact of implicit edge weight on the transfer probability. For all the candidate nodes, the one with a larger degree is more likely to be chosen.

**Mechanism to Avoid Stuck Steps.** We use an encouraging mechanism to make the random walk go further. If applying the random walk without \( \theta \), the walk steps would be stuck near the dominant nodes, and capture same nodes repetitively. Then, it can hardly go further to capture more global information. Therefore, we encourage the further step to \( v_j \) with \( \text{dis}(v_k, v_j) = 2 \). The final transfer probability \( P_{i,j} \) is calculated by normalizing the transfer weights of neighbor nodes in \( N_i \), namely

\[
P_{i,j} = \left( \frac{M_{i,j}}{\sum_{t \in N_i} M_{i,t}} \right).
\]

3.4. Local Fusion Module
The local fusion module fuses the local topology and attributes information, outputting the local embedding vector \( u_i \in \mathbb{R}^k \).

**Embedding Layer.** The node ID vector \( r_i \in \mathbb{R}^m \) and attribute vector \( a_i \in \mathbb{R}^m \) are first projected to dense vectors \( r_i' \) and \( a_i' \) through the embedding layer, i.e., \( a_i' = W_a^T a_i \), \( r_i' = W_r^T r_i \), where \( W_a \in \mathbb{R}^{n \times k_a} \) and \( W_r \in \mathbb{R}^{n \times k_r} \) are weight matrices. \( k_r \) and \( k_a \) denote the embedding sizes of the node ID and attribute respectively. It is noteworthy that the topology information is preserved by adding node ID vector. If we remove node id embedding, the nodes with same attributes will get same local embedding vectors. In this way, we only represent the attributes information of the node, without the structure information. Although the introduced node id does not contain structure information itself, it provides trainable dimension for structure information of different nodes. The actual local structure information comes from the first-order proximity in the loss function. After several epochs of training, the local structure information (in the loss function) of each node will feedback to change the original node id. The idea of utilizing node id to represent local structure has already been utilized in model ASNE [21].

**Hidden Layers.** After the embedding layer, the obtained ID vector \( r_i' \) and attribute vector \( a_i' \) are fed into a Multi-Layer Perceptron (MLP). The hidden representations for each layer are denoted as \( h_0, h_1, \ldots, h_{\ell} \), which are defined as follows,

\[
h_0 = \text{concat}(r_i', \lambda a_i'), \\
h_t = \delta_t(W_t h_{t-1} + b_t), \quad t = 1, 2, \ldots, \ell
\]

where \( \text{concat}() \) denotes the concatenation operation, \( \lambda \) adjusts the importance of attributes, \( \delta_t \) denotes the activate function, and \( \ell \) is the number of hidden layers. The final output \( h_{\ell} \) is the local embedding vector \( u_i \).

3.5. Path Aggregation Module
The global embedding vector \( u_i \) depicts the impact of high-order neighbor nodes on node \( u_i \). The path aggregation module aims to obtain the global embedding vector by updating it iteratively. The aggregation begins with an initial value, which is calculated as follows,

\[
h_{\text{con}}^i = W_a^T a_i.
\]

Then, we update the initial value by aggregating the sampled nodes. After each random walk sampling, we obtain a path \( \{s_1, \ldots, s_p\} \), where \( S_1 \) denotes the source node \( u_i \) and \( S_t \) (\( t \leq p \)) denotes the \((t-1)\)-th step. We aggregate the path nodes without \( S_1 \), namely \( S_t = \{s_2, \ldots, s_p\} \). In the following, we introduce three candidate aggregators.

**Mean Aggregator.** The first candidate aggregator function is the mean operator, which simply takes the element-wise mean of the vectors,

\[
h_{\text{agg}}^i = \text{MEAN}\{h_{\text{con}}^t, \forall v_t \in S_i\} = \frac{1}{|S_i|} \sum_{v_t \in S_i} h_{\text{con}}^t.
\]
Linear Aggregator. This aggregator is an extension to the mean aggregator by appending it with a linear transformation,

$$h_{agg} = W_L \cdot \text{MEAN}\{h^i_{con}, \forall v_t \in S_t\}.$$ 

LSTM Aggregator. We also examine a more complex aggregator based on an LSTM architecture,

$$h_{agg}^i = \text{LSTM}\{h^i_{con}, \forall v_t \in S_t\}.$$ 

Compared to the mean aggregator, LSTMs have the advantage of larger expressive capability, and they can preserve the order of a sequence. It is noteworthy that we input the path to LSTM in reverse order.

For the LSTM aggregator, we introduce an extra task to help with the training of LSTM [26]. Specifically, we aim to maximize the likelihood of predicting the next node $P(s_{next}|S_{input})$, where $S_{input} = \{s_p, \ldots, s_2\}$ represents the input sequence of LSTM, and $S_{next} = \{s_{p-1}, \ldots, s_1\}$ represents the next node sequence. For each input sequence, the goal is to predict its next node, so we use cross entropy to evaluate the loss,

$$L_1 = -\sum_s \sum_t x_{t+1} \log(\text{softmax}(W_L^T o_t)),$$

where $o_t$ denotes the output of the t-th step, and $x_{t+1}$ denotes the input of the (t+1)-th step. $W_L \in \mathbb{R}^{k_h \times n}$ is the weight matrix, where $k_h$ denotes the hidden size of LSTM. Then, we update the initial value $h^i_{con}$ with the aggregation $h_{agg}^i$ recursively, according to the following loss function,

$$L_2 = \sum_i ||h_{agg}^i - h^i_{con}||,$$

where $||x||$ denotes the 2-Norm of x. Finally, the obtained $h^i_{con}$ is the global embedding vector $u^i_t$.

3.6. Model Optimization
The embedding model is able to capture the high-order proximity, but we should also strengthen the first-order proximity and guarantee the connected nodes still close with each other in the new embedding space. Accordingly, we use the following loss function as in LINE [2],

$$L_3 = -\sum_{(v_i, v_j) \in \Omega} \log \sigma(u^T_i u^T_j) - \sum_{(v_i, v_s) \in \Omega^-} \log \sigma(-u^T_i u^T_s),$$

where $\Omega$ denotes the set of connected node pairs, $\Omega^-$ represents the set of negative sampling, and $\sigma$ represents the sigmoid function. The final loss function is calculated as follows,

$$L = (1 - \alpha)L_1 + \alpha L_2 + L_3.$$ (3)

After training the global and local embedding vectors for each node, we obtain the final embedding vector $\hat{y}_i$ by combining the two parts with weight coefficient $\beta_i$,

$$\hat{y}_i = \beta u^i_t + (1 - \beta)u^i_{con}.$$ 

4. Experimental Results
We empirically validate the effectiveness of the new approach PAN in comparison to the cutting-edge baselines, on three network analysis tasks, i.e. node classification, node clustering and link prediction. Moreover, we implement ablation experiments to validate the effectiveness of the novel random walk strategy.

| Dataset | Nodes | Edges | Attributes | Classes |
|---------|-------|-------|------------|---------|
| Cora    | 2708  | 5429  | 1433       | 7       |
| Citeseer| 3312  | 4732  | 3703       | 6       |
| Pubmed  | 19717 | 44338 | 500        | 3       |
4.1. Experimental Setup

Datasets. Three widely used attributed networks datasets are adopted, i.e. Citeseer, Cora, and Pubmed [27]. The statistics of the three datasets are summarized in Table 1.

Baseline Methods. We compare our proposal with the following baselines: (1) the methods for plain networks: Deepwalk [1], Node2vec [3], LINE [2] and DNE [20], and (2) the methods for attributed networks: TADW [6], SNE [21], CANE [7], ARNL [28], DGENE [8], VGAE [11], ARVGA [12], and AGE [10]. Among these baselines, VGAE, ARVGA, and AGE are cutting-edge GCN-based methods.

Parameter Settings. To ensure fairness, we set the final embedding dimension to 128 for all the methods and datasets. For all baselines, we set the optimal parameters as suggested in their original literatures. In our approach PANE, we use two layers of fully connected neural network in local fusion module. In specific, 256 units are applied to the first layer and 128 units are applied to the second layer. We set $\epsilon$ as 1, $\theta$ as 2, and utilize grid search to find optimal parameters for different aggregators. The initial learning rate of Adam optimizer is set to 0.0001.

Evaluation Metrics. In node classification, we use micro-average and macro-average as the evaluation metrics. To measure the performance of node clustering methods, we employ two metrics: Accuracy (ACC) and Adjusted Rand Index (ARI). For link prediction, results are evaluated with the area under curve (AUC).

Table 2. The results of node classification on Cora.

| Methods    | Micro-F1 5% | Micro-F1 15% | Micro-F1 25% | Macro-F1 5% | Macro-F1 15% | Macro-F1 25% |
|------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Deepwalk   | 0.727       | 0.789       | 0.809       | 0.695       | 0.776       | 0.798       |
| Node2vec   | 0.741       | 0.789       | 0.802       | 0.719       | 0.779       | 0.793       |
| LINE       | 0.409       | 0.493       | 0.536       | 0.365       | 0.466       | 0.512       |
| DNE        | 0.703       | 0.724       | 0.746       | 0.678       | 0.716       | 0.734       |
| TADW       | 0.673       | 0.765       | 0.788       | 0.630       | 0.720       | 0.743       |
| SNE        | 0.680       | 0.718       | 0.738       | 0.655       | 0.697       | 0.727       |
| CANE       | 0.680       | 0.746       | 0.750       | 0.612       | 0.714       | 0.731       |
| DGENE      | 0.765       | 0.814       | 0.832       | 0.754       | 0.805       | 0.823       |
| AGE        | 0.793       | 0.831       | 0.839       | 0.760       | 0.812       | 0.825       |
| PANE-Mean  | 0.804       | 0.826       | 0.833       | 0.783       | 0.801       | 0.815       |
| PANE-Linear| 0.801       | 0.825       | 0.831       | 0.773       | 0.804       | 0.814       |
| PANE-LSTM  | 0.814       | 0.838       | 0.846       | 0.789       | 0.822       | 0.829       |

4.2. Node Classification

In node classification, our goal is to classify each node into one of the multiple labels. We first randomly sample a portion (i.e. 5%, 15%, 25% in this paper) of the labeled nodes, and guarantee these selected nodes cover all classes. Then, after obtaining the network embedding, a one-vs-rest logistic regression classifier could be trained and predicts the remaining labeled nodes. We repeat this process 10 times and report the mean value of the Macro-F1 and Micro-F1 scores.

The results of node classification on three datasets are shown in Table 2, 3, and 4 respectively. As shown, on Cora and Pubmed, PANE outperforms all the baseline methods. On Citeseer dataset, PANE achieves comparable results to AGE, but still outperforms other baseline methods. In particular, PANE outperforms the random walk-based baseline methods (i.e. CANE, DGENE) on three datasets with a significant improvement.

Table 3. The results of node classification on Citeseer.

| Methods    | Micro-F1 5% | Micro-F1 15% | Micro-F1 25% | Macro-F1 5% | Macro-F1 15% | Macro-F1 25% |
|------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Deepwalk   | 0.492       | 0.559       | 0.574       | 0.448       | 0.519       | 0.530       |
| Node2vec   | 0.500       | 0.548       | 0.571       | 0.453       | 0.498       | 0.520       |
| LINE       | 0.299       | 0.333       | 0.358       | 0.264       | 0.299       | 0.324       |
| DNE        | 0.502       | 0.556       | 0.571       | 0.453       | 0.514       | 0.531       |
| TADW       | 0.553       | 0.617       | 0.642       | 0.526       | 0.587       | 0.606       |
Table 4. The results of node classification on Pubmed.

| Methods  | Micro-F1 |   | Macro-F1 |   |
|----------|----------|------------------|----------|
|          | 5%       | 15%              | 25%      | 5%       | 15%              | 25%      |
| Deepwalk | 0.718    | 0.739            | 0.762    | 0.703    | 0.725            | 0.748    |
| Node2vec | 0.749    | 0.756            | 0.763    | 0.728    | 0.736            | 0.743    |
| LINE     | 0.607    | 0.633            | 0.643    | 0.574    | 0.596            | 0.606    |
| DNE      | 0.502    | 0.556            | 0.571    | 0.453    | 0.514            | 0.531    |
| TADW     | 0.726    | 0.740            | 0.743    | 0.701    | 0.717            | 0.719    |
| SNE      | 0.747    | 0.760            | 0.764    | 0.748    | 0.763            | 0.765    |
| CANE     | 0.781    | 0.789            | 0.793    | 0.774    | 0.782            | 0.785    |
| DGENE    | 0.818    | 0.831            | 0.840    | 0.820    | 0.831            | 0.836    |
| AGE      | 0.722    | 0.755            | 0.768    | 0.715    | 0.745            | 0.756    |

4.3. Node Clustering

In node clustering, we aim to assign distinct cluster to each node. After obtaining the embedding vectors, we applied k-means algorithm to the embedding vectors of nodes to cluster them into classes. As the results of node clustering fluctuate a lot in different epochs, we regard the best score of each method as the final result.

The results are shown in Table 5. As shown, our proposal PANE consistently achieves significant improvement comparing to all the baselines on all three datasets in terms of both ARI and ACC. It verifies that PANE has the capability of modeling relationships between nodes precisely. In particular, competing with the strongest baseline AGE, our model outperforms it by 10.12% in terms of ARI and 3.76% in terms of ACC on average.

Table 5. The results of node clustering.

| Methods  | Cora | Citeseer | Pubmed |
|----------|------|----------|--------|
|          | ARI  | ACC      | ARI    | ACC    | ARI    | ACC    |
| Deepwalk | 0.336| 0.576    | 0.122  | 0.366  | 0.248  | 0.648  |
| Node2vec | 0.337| 0.589    | 0.151  | 0.420  | 0.239  | 0.655  |
| LINE     | 0.312| 0.582    | 0.064  | 0.347  | 0.207  | 0.624  |
| DNE      | 0.292| 0.531    | 0.151  | 0.410  | 0.202  | 0.607  |
| TADW     | 0.314| 0.562    | 0.275  | 0.535  | 0.154  | 0.556  |
| SNE      | 0.103| 0.394    | 0.062  | 0.311  | 0.245  | 0.651  |
| DGENE    | 0.333| 0.590    | 0.117  | 0.368  | 0.138  | 0.563  |
| AGE      | 0.473| 0.700    | 0.431  | 0.681  | 0.317  | 0.700  |
| VGAE     | 0.376| 0.603    | 0.226  | 0.510  | 0.203  | 0.622  |
| ARVGA    | 0.375| 0.633    | 0.355  | 0.622  | 0.176  | 0.596  |
| ANRL     | 0.310| 0.556    | 0.337  | 0.612  | 0.265  | 0.662  |
| PANE-Mean| 0.521| 0.732    | **0.447**| 0.688  | **0.362**| 0.727  |
| PANE-Linear| **0.532**| **0.745**| 0.442  | 0.687  | 0.352  | 0.721  |
| PANE-LSTM| 0.459| 0.693    | 0.415  | 0.668  | 0.312  | 0.696  |
4.4. Link Prediction
In link prediction, we aim to assess the ability of node embedding by reconstructing the link between nodes. We follow the widely adopted way. For each dataset, the edges are randomly divided into three groups. 85%, 5% and 10% of the edges are utilized in training, validation (hyper-parameters tuning), and performance testing, respectively. As the test/validation set contains only positive instances, we randomly sample the same number of non-existing links as negative instances, and rank both positive and negative instances according to the prediction function. For each network, experiments are repeated 10 times on 10 different random edge partitions, and the average performances are recorded.

The results are shown in Table 6. As shown, compared with the baseline methods, our proposal PANE outperforms the baselines on all three datasets. In particular, compared with the strongest baseline AGE, PANE outperforms it by 1.27% on Cora, 2.19% on Citeseer, and 8.70% on Pubmed.

Table 6. The results of link prediction.

| Methods | Cora | Citeseer | Pubmed |
|---------|------|----------|--------|
| Deepwalk  | 0.831 | 0.805 | 0.844 |
| Node2vec | 0.888 | 0.837 | 0.883 |
| LINE     | 0.869 | 0.807 | 0.847 |
| TADW     | 0.890 | 0.945 | 0.902 |
| SNE      | 0.922 | 0.904 | 0.886 |
| CANE     | 0.932 | 0.945 | 0.924 |
| DGENE    | 0.904 | 0.857 | 0.931 |
| AGE      | 0.944 | 0.956 | 0.873 |
| VGAE     | 0.916 | 0.896 | 0.944 |
| ARVGA    | 0.920 | 0.914 | 0.905 |
| ANRL     | 0.871 | 0.930 | 0.915 |
| PANE-Mean  | **0.956** | **0.977** | **0.935** |
| PANE-Linear | 0.946 | 0.971 | **0.938** |
| PANE-LSTM | 0.938 | 0.964 | 0.949 |

5. Conclusion
To better utilize the high order information of attributed networks, we propose a path aggregation model for attributed network embedding (PANE) with a tweaked random walk. PANE strengthens the leveraging of high order information from two aspects. On one hand, PANE strengthens the path sampling by proposing a tweaked random walk strategy, which can capture the nodes with more high order information preferentially. On the other hand, PANE strengthens the aggregation process by introducing a path-aggregation framework, which is rarely used in the field of attributed network embedding. By fully leveraging the rich high order information, PANE achieves competitive performances and outperforms the cutting-edge baselines in most cases.

Acknowledgments
This work is supported by the National Natural Science Foundation of China (62002396, 61902438, 61902439, U1811264, U19112031), Natural Science Foundation of Guangdong Province under Grant (2020A1515011251, 2019A1515011704, 2019A1515011159), Guangdong Basic and Applied Basic Research Foundation (2019B1515130001), National Science Foundation for Post-Doctoral Scientists of China under Grant (2018M643307, 2019M663237), the Major Science and Technology Research Programs of Zhongshan City (2019B2006, 2019A40027, 2021A1003).

References
[1] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD, 2014, pp. 701–710.
[2] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: WWW 2015, pp. 1067–1077.
[3] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks. 22nd ACM SIGKDD 2016, pp. 855–864.

[4] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the 22nd ACM SIGKDD 2016, pp. 1225–1234.

[5] K. Tu, P. Cui, X. Wang, P. S. Yu, W. Zhu, Deep recursive network embedding with regular equivalence, in: Proceedings of the 24th ACM SIGKDD 2018, pp. 2357–2366.

[6] C. Yang, Z. Liu, D. Zhao, M. Sun, E. Y. Chang, Network representation learning with rich text information., in: IJCAI, Vol. 2015, 2015, pp. 2111–2117.

[7] C. Tu, H. Liu, Z. Liu, M. Sun, Cane: Context-aware network embedding for relation modeling, in: Proceedings of the 55th ACL (Volume 1: Long Papers), 2017, pp. 1722–1731.

[8] J. Liu, N. Li, Z. He, Network embedding with dual generation tasks. IJCAI 2019, 2019.

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

[10] G. Cui, J. Zhou, C. Yang, Z. Liu, Adaptive graph encoder for attributed graph embedding, in: Proceedings of the 26th ACM SIGKDD, 2020, pp. 976–985.

[11] T. N. Kipf, M. Welling, Variational graph auto-encoders. NIPS 2016.

[12] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, C. Zhang, Adversarially regularized graph autoencoder for graph embedding, arXiv preprint arXiv:1804.04407.

[13] D. He, L. Zhai, Z. Li, L. Yang, D. Jin, Y. Huang, P. S. Yu, Adversarial mutual information learning for network embedding, in: Proceedings of IJCAI, 2020, pp. 3321–3327.

[14] S. Bandyopadhyay, N. Lokesh, M. N. Murty, Outlier aware network embedding for attributed networks. AAAI 2019, Vol. 33, 2019, pp. 12–19.

[15] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, science 290 (5500) (2000) 2323–2326.

[16] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, Advances in Neural Information Processing Systems 14 (6) (2001) 585–591.

[17] X. Huang, J. Li, X. Hu, Accelerated attributed network embedding. SIAM, 2017, pp. 633–641.

[18] X. Fu, J. Zhang, Z. Meng, I. King, Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding, in: Proceedings of The Web Conference 2020, 2020, pp. 2331–2341.

[19] D. Chen, L. Jacob, J. Mairal, Convolutional kernel networks for graphstructured data, in: International Conference on Machine Learning, PMLR, 2020, pp. 1576–1586.

[20] X. Sun, Z. Song, J. Dong, Y. Yu, C. Plant, C. Böhm, Network structure and transfer behaviors embedding via deep prediction model, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 5041–5048.

[21] L. Liao, X. He, H. Zhang, T.-S. Chua, Attributed social network embedding. IEEE Transactions on Knowledge and Data Engineering 30 (12) (2018) 2257–2270.

[22] C. Tu, W. Zhang, Z. Liu, M. Sun, et al., Max-margin deepwalk: Discriminative learning of network representation., in: IJCAI, Vol. 2016, 2016, pp. 3889–3895.

[23] S. Pan, J. Wu, X. Zhu, C. Zhang, Y. Wang, Tri-party deep network representation, Network 11 (9) (2016) 12.

[24] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in neural information processing systems, 2017, pp. 1024–1034.

[25] W. Xiao, H. Zhao, V. W. Zheng, Y. Song, Vertex-reinforced random walk for network embedding. SIAM, 2020, pp. 595–603.

[26] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781.

[27] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, T. Eliassi-Rad, Collective classification in network data, AI magazine 29 (3) (2008) 93–93.
[28] 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: IJCAI, Vol. 18, 2018, pp. 3155–3161.