Attributed Network Embedding for Learning in a Dynamic Environment

Jundong Li
Arizona State University
jundongl@asu.edu

Harsh Dani
Arizona State University
hdani@asu.edu

Xia Hu
Texas A&M University
hu@cse.tamu.edu

Jiliang Tang
Michigan State University
tangjili@msu.edu

Yi Chang
Huawei Research America
yichang@acm.org

Huan Liu
Arizona State University
huan.liu@asu.edu

ABSTRACT

Network embedding leverages the node proximity manifested to learn a low-dimensional node vector representation. The learned embeddings could advance various learning tasks such as node classification, network clustering, and link prediction. Most, if not all, of the existing work, is overwhelmingly performed in the context of plain and static networks. Nonetheless, in reality, network structure often evolves over time with addition/deletion of links and nodes. Also, a vast majority of real-world networks are associated with a rich set of node attributes, and their attribute values are also naturally changing, with the emerging of new content and the fading of old content. These changing characteristics motivate us to seek an effective embedding representation to capture network and attribute evolving patterns, which is of fundamental importance for learning in a dynamic environment. To our best knowledge, we are the first to tackle this problem with the following two challenges: (1) the inherently correlated network and node attributes could be noisy and incomplete, it necessitates a robust consensus representation to capture their individual properties and correlations; (2) the embedding learning needs to be performed in an online fashion to adapt to the changes accordingly. In this paper, we tackle this problem by proposing a novel dynamic attributed network embedding framework - DANE. In particular, DANE provides an offline method for a consensus embedding first and then leverages matrix perturbation theory to maintain the freshness of the end embedding results in an online manner. We perform extensive experiments on both synthetic and real attributed networks to corroborate the effectiveness and efficiency of the proposed framework.

KEYWORDS

Dynamic Networks; Attributed Networks; Network Embedding

ACM Reference format:
Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 1997. Attributed Network Embedding for Learning in a Dynamic Environment. In *Proceedings of ACM WOODSTOCK conference, El Paso, Texas, USA, 1997 (WOODSTOCK'97),* 10 pages.
DOI: 10.475/123_4

WOODSTOCK'97, El Paso, Texas, USA

DOI: 10.475/123_4

1 INTRODUCTION

Attributed networks are ubiquitous in a myriad of high impact domains, ranging from social media networks, academic networks, to protein-protein interaction networks. In contrast to conventional plain networks where only pairwise node dependencies are observed, nodes in attributed networks are often affiliated with a rich set of attributes. For example, in scientific collaboration networks, researchers collaborate and are distinct from others by their unique research interests and foci; in social networks, users interact and communicate with others and also post personalized content. It has been widely studied and received that there exhibits a strong correlation among the attributes of linked nodes [27, 29]. The root cause of the correlations can be attributed to social influence and homophily effect in social science theories [22, 23]. Also, many real-world applications, such as community detection, sentiment analysis, and social spammer detection [15, 31, 41], have shown significant improvements by modeling such correlations.

Network embedding [9, 26, 33] has attracted a surge of research attention in recent years. The basic idea is to preserve the node proximity in the embedded Euclidean space, based on which the performance of various network mining tasks such as node classification [3, 5], community detection [34, 41], and link prediction [21, 39] can be enhanced. However, a vast majority of existing work are predominately designed for plain networks. They inevitably ignore the node attributes that could be potentially complementary in learning better embedding representations, especially when the network suffers from high sparsity. In addition, a fundamental assumption behind existing network embedding methods is that networks are static and given a prior. Nonetheless, most real-world networks are intrinsically dynamic with addition/deletion of edges and nodes; examples include co-author relations between scholars in an academic network and friendships among users in a social network. Meanwhile, similar as network structure, node attributes also change naturally such that new content patterns may emerge and outdated content patterns will fade. For example, humanitarian and disaster relief related topics become popular on social media sites after the earthquakes as users continuously post related

content. Consequently, other topics may receive less public interests. In this paper, we refer this kind of networks with both network and node attribute value changes as *dynamic attributed networks*.

Despite the widespread of dynamic attributed networks in real-world applications, the study in analyzing and mining these networks are rather limited. One natural question to ask is when attributed networks evolve, how to correct and adjust the staleness of the end embedding results for network analysis, which will shed light on the understanding of their evolving nature. However, dynamic attributed network embedding remains as a daunting task, mainly because of the following reasons: (1) Even though network topology and node attributes are two distinct data representations, they are inherently correlated. In addition, the raw data representations could be noisy and even incomplete, individually. Hence, it is of paramount importance to seek a noise-resilient consensus embedding to capture their individual properties and correlations; (2) Applying offline embedding methods from scratch at each time step is time-consuming and cannot seize the emerging patterns timely. It necessitates the design of an efficient online algorithm that can give embedding representations promptly.

To tackle the aforementioned challenges, we propose a novel embedding framework for dynamic attributed networks. The main contributions can be summarized as follows:

- ***Problem Formulations***: we formally define the problem of dynamic attributed network embedding. The key idea is to initiate an offline model at the very beginning, based on which an online model is presented to maintain the freshness of the end attributed network embedding results.
- ***Algorithms and Analysis***: we propose a novel framework - DANE for dynamic attributed network embedding. Specifically, we introduce an offline embedding method as a base model to preserve node proximity in terms of both network structure and node attributes for a consensus embedding representation in a robust way. Then to timely obtain an updated embedding representation when both network structure and attributes drift, we present an online model to update the consensus embedding with matrix perturbation theory. We also theoretically analyze its time complexity and show its superiority over offline methods.
- ***Evaluations***: we perform extensive experiments on both synthetic and real-world attributed networks to corroborate the efficacy in terms of two network mining tasks (both unsupervised and supervised). Also, we show its efficiency by comparing it with other baseline methods and its offline counterpart. In particular, our experimental results show that the proposed method outperforms the best competitors in terms of both clustering and classification performance. Most importantly, it is much more efficient than existing competitive offline embedding methods.

The rest of this paper is organized as follows. The problem statement of dynamic attributed network embedding is introduced in Section 2. Section 3 presents the proposed framework DANE with analysis. Experiments on synthetic and real datasets are presented in Section 4 with discussions. Section 5 briefly reviews related work. Finally, Section 6 concludes the paper and visions the future work.

2 PROBLEM DEFINITION

We first summarize some notations used in this paper. Following the commonly used notations, we use bold uppercase characters for matrices (e.g., $\mathbf{A}$), bold lowercase characters for vectors (e.g., $\mathbf{a}$), normal lowercase characters for scalars (e.g., a). Also we represent the i-th row of matrix $\mathbf{A}$ as $\mathbf{A}(i,:)$, the j-th column as $\mathbf{A}(:,j)$, the (i,j)-th entry as $\mathbf{A}(i,j)$, transpose of $\mathbf{A}$ as $\mathbf{A}'$, trace of $\mathbf{A}$ as $tr(\mathbf{A})$ if it is a square matrix. $\mathbf{1}$ denotes a vector whose elements are all 1 and $\mathbf{I}$ denotes the identity matrix. The main symbols used throughout this paper are listed in Table 1.

Notations	Definitions or Descriptions
$\mathcal{G}^{(t)}$	attributed network at time step t
$\mathcal{G}^{(t+1)}$	attributed network at time step $t+1$
$\mathbf{A}^{(t)}$	adjacency matrix for the network structure in $\mathcal{G}^{(t)}$
$\mathbf{X}^{(t)}$	attribute information in $\mathcal{G}^{(t)}$
$\mathbf{A}^{(t+1)}$	adjacency matrix for the network structure in $\mathcal{G}^{(t+1)}$
$\mathbf{X}^{(t+1)}$	attribute information in $\mathcal{G}^{(t+1)}$
$\Delta\mathbf{A}$	change of adjacency matrix between time steps t and $t+1$
$\Delta\mathbf{X}$	change of attribute values between time steps t and $t+1$
n	number of instances (nodes) in $\mathcal{G}^{(t)}$
d	number of attributes in $\mathcal{G}^{(t)}$
k	embedding dimension for network structure or attributes
l	final consensus embedding dimension

Table 1: Symbols.

Let $\mathcal{U}^{(t)} = \{u_1, u_2, ..., u_n\}$ denote a set of n nodes in the attributed network $\mathcal{G}^{(t)}$ at time step t. We use the adjacency matrix $\mathbf{A}^{(t)} \in \mathbb{R}^{n\times n}$ to represent the network structure of $\mathcal{U}^{(t)}$. In addition, we assume that nodes are affiliated with d-dimensional attributes $\mathcal{F} = \{f_1, f_2, ..., f_d\}$ and $\mathbf{X}^{(t)} \in \mathbb{R}^{n\times d}$ denotes node attributes. At the following time step, the attributed network is characterized with both topology and content drift such that new/old edges and nodes may be included/deleted, and node attribute values could also change. We use $\Delta\mathbf{A}$ and $\Delta\mathbf{X}$ to denote the network and attribute value changes between two consecutive time step t and time step $t+1$, respectively. Following the settings of [36], and for the ease of presentation, we consider the number of nodes is constant over time, but our method can be naturally extended to deal with node addition/deletion scenarios. As mentioned earlier, node attributes are complementary in mitigating the network sparsity for better embedding representations. Nonetheless, employing offline embedding methods repeatedly in a dynamic environment is time-consuming and cannot seize the emerging/fading patterns promptly, especially when the networks are of large-scale. Therefore, developing an efficient online embedding algorithm upon an offline model is fundamentally important for dynamic network analysis, and also could benefit many real-wold applications. Formally, we define the dynamic attributed embedding problem as two sub-problems as follows. The work flow of the proposed framework DANE is shown in Figure 1.

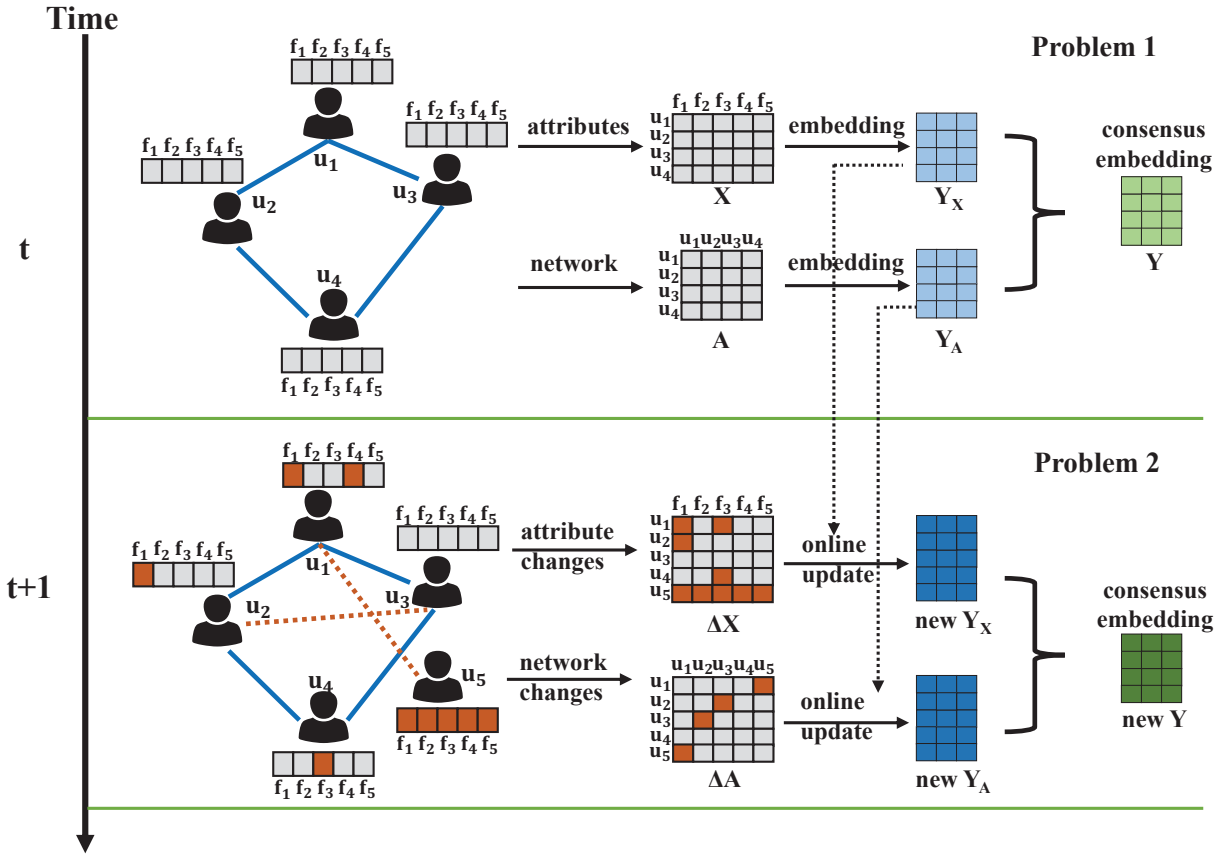


Figure 1: An illustration of the proposed framework dynamic attributed network embedding framework - DANE. At time step t, DANE performs spectral embedding on network structure A and node attributes X, and obtain two embeddings Y_A and Y_X. Afterwards, DANE maximizes their correlation for a consensus embedding representation Y. At the following time step $t+1$, the network is characterized by both topology structure and attribute value changes ΔA and ΔX (the changes are highlighted in orange). DANE leverages matrix perturbation theory to update Y_A and Y_X, and give the new consensus embedding Y.

Problem 1. *The offline model of DANE at time step t: given network topology $\mathbf{A}^{(t)}$ and node attributes $\mathbf{X}^{(t)}$; output attributed network embedding $\mathbf{Y}^{(t)}$ for all nodes.*

Problem 2. *The online model of DANE at time step $t+1$: given network topology $\mathbf{A}^{(t+1)}$ and node attributes $\mathbf{X}^{(t+1)}$, and intermediate embedding results at time step t; output attributed network embedding $\mathbf{Y}^{(t+1)}$ for all nodes.*

3 THE PROPOSED FRAMEWORK - DANE

In this section, we first present an offline model that works in a static setting to tackle the Problem 1 in finding a consensus embedding representation. Then to tackle the Problem 2, we introduce an online model that provides a fast solution to update the consensus embedding on the fly. At the end, we analyze the computational complexity the online model and show its superiority over the offline model.

3.1 DANE: Offline Model

Network topology and node attributes in attributed networks are presented in different representations. Typically, either of these two representations could be *incomplete* and *noisy*, presenting great challenges to learn embedding. For example, social networks are very sparse as a sheer amount of users only have a limited number of links [1]. Thus, network embedding could be jeopardized as links are inadequate to provide enough node proximity information. Fortunately, rich node attributes are readily available and could be potentially helpful to mitigate the network sparsity in finding better embeddings. Hence, it is more desired to make these two representations compensate each other for a consensus embedding. However, as mentioned earlier, both representations could be noisy and the existence of noise could degenerate the learning of consensus embedding. Hence, it motivates us to reduce the noise of these two raw data representations before learning consensus embedding.

Let $\mathbf{A}^{(t)} \in \mathbb{R}^{n \times n}$ be the adjacency matrix of the attributed network at time step t and $\mathbf{D}_\mathbf{A}$ be the diagonal matrix with $\mathbf{D}_\mathbf{A}^{(t)}(i,i) =$

$\sum_{j=1}^{n} \mathbf{A}^{(t)}(i,j)$, then $\mathbf{L}_{\mathbf{A}}^{(t)} = \mathbf{D}_{\mathbf{A}}^{(t)} - \mathbf{A}^{(t)}$ is a Laplacian matrix. According to spectral theory [4, 37], by mapping each node in the network to a k-dimensional embedded space, i.e., $\mathbf{y}_i \in \mathbb{R}^k$ ($k \ll n$), the noise in the network can be substantially reduced. A rational choice of the embedding $\mathbf{Y}_{\mathbf{A}}^{(t)} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n]' \in \mathbb{R}^{n \times k}$ is to minimize the loss $\frac{1}{2}\sum_{i,j} \mathbf{A}^{(t)}(i,j)||\mathbf{y}_i - \mathbf{y}_j||_2^2$. It ensures that connected nodes are close to each other in the embedded space. In this case, the problem boils down to solving the following generalized eigen-problem $\mathbf{L}_{\mathbf{A}}^{(t)}\mathbf{a} = \lambda \mathbf{D}_{\mathbf{A}}^{(t)}\mathbf{a}$. Let $\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n$ be the eigenvectors of the corresponding eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$. It is easy to verify that $\mathbf{1}$ is the only eigenvector for the eigenvalue $\lambda_1 = 0$. Then the k-dimensional embedding $\mathbf{Y}_{\mathbf{A}}^{(t)} \in \mathbb{R}^{n \times k}$ of the network structure is given by the top-k eigenvectors starting from $\mathbf{a}_2$, i.e., $\mathbf{Y}_{\mathbf{A}}^{(t)} = [\mathbf{a}_2, ..., \mathbf{a}_k, \mathbf{a}_{k+1}]$. For the ease of presentation, in the following parts of the paper, we refer these k eigenvectors and their eigenvalues as the top-k eigenvectors and eigenvalues, respectively. Akin to the network structure, noise in the node attributes can be reduced in a similar fashion. Specifically, we first normalize attributes of each node and obtain the cosine similarity matrix $\mathbf{W}^{(t)}$. Afterwards, we obtain the top-k eigenvectors $\mathbf{Y}_{\mathbf{X}}^{(t)} = [\mathbf{b}_2, ..., \mathbf{b}_{k+1}]$ of the generalized eigen-problem corresponding to $\mathbf{W}^{(t)}$.

The noisy data problem is resolved by finding two intermediate embeddings $\mathbf{Y}_{\mathbf{A}}^{(t)}$ and $\mathbf{Y}_{\mathbf{X}}^{(t)}$. We now take advantage of them to seek a consensus embedding. However, since they are obtained individually, these two embeddings may not be compatible and in the worst case, they may be independent of each other. To capture their interdependency and to make them compensate each other, we propose to maximize their correlations (or equivalently minimize their disagreements) [14]. In particular, we seek two projection vectors $\mathbf{p}_{\mathbf{A}}^{(t)}$ and $\mathbf{p}_{\mathbf{X}}^{(t)}$ such that the correlation of $\mathbf{Y}_{\mathbf{A}}^{(t)}$ and $\mathbf{Y}_{\mathbf{X}}^{(t)}$ is maximized after projection. It is equivalent to solving the following optimization problem:

$$\begin{aligned} &\max_{\mathbf{p}_{\mathbf{A}}^{(t)}, \mathbf{p}_{\mathbf{X}}^{(t)}} \mathbf{p}_{\mathbf{A}}^{(t)'}\mathbf{Y}_{\mathbf{A}}^{(t)'}\mathbf{Y}_{\mathbf{A}}^{(t)}\mathbf{p}_{\mathbf{A}}^{(t)} + \mathbf{p}_{\mathbf{A}}^{(t)'}\mathbf{Y}_{\mathbf{A}}^{(t)'}\mathbf{Y}_{\mathbf{X}}^{(t)}\mathbf{p}_{\mathbf{X}}^{(t)} \\ &+ \mathbf{p}_{\mathbf{X}}^{(t)'}\mathbf{Y}_{\mathbf{X}}^{(t)'}\mathbf{Y}_{\mathbf{A}}^{(t)}\mathbf{p}_{\mathbf{A}}^{(t)} + \mathbf{p}_{\mathbf{X}}^{(t)'}\mathbf{Y}_{\mathbf{X}}^{(t)'}\mathbf{Y}_{\mathbf{X}}^{(t)}\mathbf{p}_{\mathbf{X}}^{(t)}. \\ &\text{s.t.} \quad \mathbf{p}_{\mathbf{A}}^{(t)'}\mathbf{Y}_{\mathbf{A}}^{(t)'}\mathbf{Y}_{\mathbf{A}}^{(t)}\mathbf{p}_{\mathbf{A}}^{(t)} + \mathbf{p}_{\mathbf{X}}^{(t)'}\mathbf{Y}_{\mathbf{X}}^{(t)'}\mathbf{Y}_{\mathbf{X}}^{(t)}\mathbf{p}_{\mathbf{X}}^{(t)} = 1. \end{aligned} \tag{1}$$

Let γ be the Lagrange multiplier for the constraint, by setting the derivative of the Lagrange function w.r.t. $\mathbf{p}_{\mathbf{A}}^{(t)}$ and $\mathbf{p}_{\mathbf{X}}^{(t)}$ to zero, we obtain the optimal solution for $[\mathbf{p}_{\mathbf{A}}^{(t)}; \mathbf{p}_{\mathbf{X}}^{(t)}]$, which corresponds to the eigenvector of the following generalized eigen-problem:

$$\begin{bmatrix} \mathbf{Y}_{\mathbf{A}}^{(t)'}\mathbf{Y}_{\mathbf{A}}^{(t)} & \mathbf{Y}_{\mathbf{A}}^{(t)'}\mathbf{Y}_{\mathbf{X}}^{(t)} \\ \mathbf{Y}_{\mathbf{X}}^{(t)'}\mathbf{Y}_{\mathbf{A}}^{(t)} & \mathbf{Y}_{\mathbf{X}}^{(t)'}\mathbf{Y}_{\mathbf{X}}^{(t)} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{\mathbf{A}}^{(t)} \\ \mathbf{p}_{\mathbf{X}}^{(t)} \end{bmatrix} = \gamma \begin{bmatrix} \mathbf{Y}_{\mathbf{A}}^{(t)'}\mathbf{Y}_{\mathbf{A}}^{(t)} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}_{\mathbf{X}}^{(t)'}\mathbf{Y}_{\mathbf{X}}^{(t)} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{\mathbf{A}}^{(t)} \\ \mathbf{p}_{\mathbf{X}}^{(t)} \end{bmatrix}. \tag{2}$$

As a result, to obtain a consensus embedding representation from $\mathbf{Y}_{\mathbf{A}}$ and $\mathbf{Y}_{\mathbf{X}}$, we could take the top-l eigenvectors of the above generalized eigen-problem and stack these top-l eigenvectors together. Suppose the projection matrix $\mathbf{P}^{(t)} \in \mathbb{R}^{2k \times l}$ is the concatenated top-l eigenvectors, the final consensus embedding representation can be computed as $\mathbf{Y}^{(t)} = [\mathbf{Y}_{\mathbf{A}}^{(t)}, \mathbf{Y}_{\mathbf{X}}^{(t)}] \times \mathbf{P}^{(t)}$.

3.2 Online Model of DANE

More often than not, attributed networks often exhibit high dynamics. For example, in social media sites, social relations are continuously evolving, and user posting behaviors may also evolve accordingly. It raises challenges to the existing offline embedding methods as they have to rerun at each time step which is time-consuming and is not scalable to large networks. Therefore, it is important to build an efficient online embedding algorithm which gives an informative embedding representation on the fly.

The proposed online embedding model is motivated by the observation that most of real-world networks, with no exception for attributed networks, often evolve smoothly in the temporal dimension between two consecutive time steps [2, 10]. Hence, we use $\Delta\mathbf{A}$ and $\Delta\mathbf{X}$ to denote the perturbation of network structure and node attributes between two consecutive time steps t and $t+1$, respectively. With these, the diagonal matrix and Laplacian matrix of $\mathbf{A}$ and $\mathbf{X}$ also evolve smoothly such that:

$$\begin{aligned} \mathbf{D}_{\mathbf{A}}^{(t+1)} &= \mathbf{D}_{\mathbf{A}}^{(t)} + \Delta\mathbf{D}_{\mathbf{A}}, \quad \mathbf{L}_{\mathbf{A}}^{(t+1)} = \mathbf{L}_{\mathbf{A}}^{(t)} + \Delta\mathbf{L}_{\mathbf{A}}, \\ \mathbf{D}_{\mathbf{X}}^{(t+1)} &= \mathbf{D}_{\mathbf{X}}^{(t)} + \Delta\mathbf{D}_{\mathbf{X}}, \quad \mathbf{L}_{\mathbf{X}}^{(t+1)} = \mathbf{L}_{\mathbf{X}}^{(t)} + \Delta\mathbf{L}_{\mathbf{X}}. \end{aligned} \tag{3}$$

As discussed in the previous subsection, the problem of attributed network embedding in an offline setting boils down to solving generalized eigen-problems. In particular, offline model focuses on finding the top eigenvectors corresponding to the smallest eigenvalues of the generalized eigen-problems. Therefore, the core idea to enable online update of the embeddings is to develop an efficient way to update the top eigenvectors and eigenvalues. Otherwise, we have to perform generalized eigen-decomposition each time step, which is not practical due to its high time complexity.

Without loss of generality, we use the network topology as an example to illustrate the proposed algorithm for online embedding. By the matrix perturbation theory [30], we have the following equation in embedding the network structure at the new time step:

$$(\mathbf{L}_{\mathbf{A}}^{(t)} + \Delta\mathbf{L}_{\mathbf{A}})(\mathbf{a} + \Delta\mathbf{a}) = (\lambda + \Delta\lambda)(\mathbf{D}_{\mathbf{A}}^{(t)} + \Delta\mathbf{D}_{\mathbf{A}})(\mathbf{a} + \Delta\mathbf{a}). \tag{4}$$

For a specific eigen-pair $(\lambda_i, \mathbf{a}_i)$, we have the following equation:

$$(\mathbf{L}_{\mathbf{A}}^{(t)} + \Delta\mathbf{L}_{\mathbf{A}})(\mathbf{a}_i + \Delta\mathbf{a}_i) = (\lambda_i + \Delta\lambda_i)(\mathbf{D}_{\mathbf{A}}^{(t)} + \Delta\mathbf{D}_{\mathbf{A}})(\mathbf{a}_i + \Delta\mathbf{a}_i). \tag{5}$$

The problem now is how to compute the change of the i-th eigenpair $(\Delta\mathbf{a}_i, \Delta\lambda_i)$ by taking advantage of the small perturbation matrices $\Delta\mathbf{D}$ and $\Delta\mathbf{L}$.

A - Computing the change of eigenvalue $\Delta\lambda_i$.
By expanding the above equation, we have:

$$\begin{aligned} &\mathbf{L}_{\mathbf{A}}^{(t)}\mathbf{a}_i + \Delta\mathbf{L}_{\mathbf{A}}\mathbf{a}_i + \mathbf{L}_{\mathbf{A}}^{(t)}\Delta\mathbf{a}_i + \Delta\mathbf{L}_{\mathbf{A}}\Delta\mathbf{a}_i \\ =&\lambda_i\mathbf{D}_{\mathbf{A}}^{(t)}\mathbf{a}_i + \lambda_i\Delta\mathbf{D}_{\mathbf{A}}\mathbf{a}_i + \Delta\lambda_i\mathbf{D}_{\mathbf{A}}^{(t)}\mathbf{a}_i + \Delta\lambda_i\Delta\mathbf{D}_{\mathbf{A}}\mathbf{a}_i \\ +&(\lambda_i\mathbf{D}_{\mathbf{A}}^{(t)} + \lambda_i\Delta\mathbf{D}_{\mathbf{A}} + \Delta\lambda_i\mathbf{D}_{\mathbf{A}}^{(t)} + \Delta\lambda_i\Delta\mathbf{D}_{\mathbf{A}})\Delta\mathbf{a}_i. \end{aligned} \tag{6}$$

The higher order terms, i.e., $\Delta\lambda_i\Delta\mathbf{D}_{\mathbf{A}}\mathbf{a}_i$, $\lambda_i\Delta\mathbf{D}_{\mathbf{A}}\Delta\mathbf{a}_i$, $\Delta\lambda_i\mathbf{D}_{\mathbf{A}}^{(t)}\Delta\mathbf{a}_i$ and $\Delta\lambda_i\Delta\mathbf{D}_{\mathbf{A}}\Delta\mathbf{a}_i$ can be removed as they have limited effects on the accuracy of a generalized eigen-systems [12]. By using the fact that $\mathbf{L}_{\mathbf{A}}^{(t)}\mathbf{a}_i = \lambda_i\mathbf{D}_{\mathbf{A}}^{(t)}\mathbf{a}_i$, we have the following formulation:

$$\Delta\mathbf{L}_{\mathbf{A}}\mathbf{a}_i + \mathbf{L}_{\mathbf{A}}^{(t)}\Delta\mathbf{a}_i = \lambda_i\Delta\mathbf{D}_{\mathbf{A}}\mathbf{a}_i + \Delta\lambda_i\mathbf{D}_{\mathbf{A}}^{(t)}\mathbf{a}_i + \lambda_i\mathbf{D}_{\mathbf{A}}^{(t)}\Delta\mathbf{a}_i. \tag{7}$$

Multiplying both sides with $\mathbf{a}_i'$, we now have:

$$\mathbf{a}_i'\Delta\mathbf{L}_\mathbf{A}\mathbf{a}_i + \mathbf{a}_i'\mathbf{L}_\mathbf{A}^{(t)}\Delta\mathbf{a}_i = \lambda_i\mathbf{a}_i'\Delta\mathbf{D}_\mathbf{A}\mathbf{a}_i + \Delta\lambda_i\mathbf{a}_i'\mathbf{D}_\mathbf{A}^{(t)}\mathbf{a}_i + \lambda_i\mathbf{a}_i'\mathbf{D}_\mathbf{A}^{(t)}\Delta\mathbf{a}_i. \tag{8}$$

Since both the Laplacian matrix $\mathbf{L}_\mathbf{A}^{(t)}$ and the diagonal matrix $\mathbf{D}_\mathbf{A}^{(t)}$ are symmetric, we have:

$$\mathbf{a}_i'\mathbf{L}_\mathbf{A}^{(t)}\Delta\mathbf{a}_i = \lambda_i\mathbf{a}_i'\mathbf{D}_\mathbf{A}^{(t)}\Delta\mathbf{a}_i. \tag{9}$$

Therefore, Eq. (8) can be reformulated as follows:

$$\mathbf{a}_i'\Delta\mathbf{L}_\mathbf{A}\mathbf{a}_i = \lambda_i\mathbf{a}_i'\Delta\mathbf{D}_\mathbf{A}\mathbf{a}_i + \Delta\lambda_i\mathbf{a}_i'\mathbf{D}_\mathbf{A}^{(t)}\mathbf{a}_i. \tag{10}$$

Through this, the variation of eigenvalue, i.e., $\Delta\lambda_i$, is:

$$\Delta\lambda_i = \frac{\mathbf{a}_i'\Delta\mathbf{L}_\mathbf{A}\mathbf{a}_i - \lambda_i\mathbf{a}_i'\Delta\mathbf{D}_\mathbf{A}\mathbf{a}_i}{\mathbf{a}_i'\mathbf{D}_\mathbf{A}^{(t)}\mathbf{a}_i}. \tag{11}$$

Theorem 3.1. *In the generalized eigen-problem* $\mathbf{A}\mathbf{v} = \lambda\mathbf{B}\mathbf{v}$*, if* $\mathbf{A}$ *and* $\mathbf{B}$ *are both Hermitian matrices and* $\mathbf{B}$ *is a positive-semidefinite matrix, the eigenvalue* λ *are real; and eigenvectors* $\mathbf{v}_j$ $(i \neq j)$ *are* $\mathbf{B}$*-orthogonal such that* $\mathbf{v}_i'\mathbf{B}\mathbf{v}_j = 0$ *and* $\mathbf{v}_i'\mathbf{B}\mathbf{v}_i = 1$ *[25].*

Corollary 3.2. $\mathbf{a}_i'\mathbf{D}_\mathbf{A}^{(t)}\mathbf{a}_i = 1$ *and* $\mathbf{a}_i'\mathbf{D}_\mathbf{A}^{(t)}\mathbf{a}_j = 0$ $(i \neq j)$.

Proof. Both $\mathbf{D}_\mathbf{A}^{(t)}$ and $\mathbf{L}_\mathbf{A}^{(t)}$ are symmetric and are also Hermitian matrices. Meanwhile, the Laplacian matrix $\mathbf{L}_\mathbf{A}^{(t)}$ is a positive-definite matrix, which completes the proof. □

Therefore, the variation of the eigenvalue λ_i is as follows:

$$\Delta\lambda_i = \mathbf{a}_i'\Delta\mathbf{L}_\mathbf{A}\mathbf{a}_i - \lambda_i\mathbf{a}_i'\Delta\mathbf{D}_\mathbf{A}\mathbf{a}_i. \tag{12}$$

B - Computing the change of eigenvector $\Delta\mathbf{a}_i$.

As network structure often evolves smoothly between two continuous time steps, we assume that the perturbation of the eigenvectors $\Delta\mathbf{a}_i$ lies in the column space that is composed by the top-k eigenvectors at time step t such that $\Delta\mathbf{a}_i = \sum_{j=2}^{k+1}\alpha_{ij}\mathbf{a}_j$, where α_{ij} is a weight indicating the contribution of the j-th eigenvector $\mathbf{a}_j$ in approximating the new i-th eigenvector. Next, we show how to determine these weights such that the perturbation $\Delta\mathbf{a}_i$ can be estimated.

By plugging $\Delta\mathbf{a}_i = \sum_{j=2}^{k+1}\alpha_{ij}\mathbf{a}_j$ into Eq. (7) and using the fact that $\mathbf{L}_\mathbf{A}^{(t)}\sum_{j=2}^{k+1}\alpha_{ij}\mathbf{a}_j = \mathbf{D}_\mathbf{A}^{(t)}\sum_{j=2}^{k+1}\alpha_{ij}\lambda_j\mathbf{a}_j$, we obtain the following:

$$\Delta\mathbf{L}_\mathbf{A}\mathbf{a}_i + \mathbf{D}_\mathbf{A}^{(t)}\sum_{j=2}^{k+1}\alpha_{ij}\lambda_j\mathbf{a}_j = \lambda_i\Delta\mathbf{D}_\mathbf{A}\mathbf{a}_i + \Delta\lambda_i\mathbf{D}_\mathbf{A}^{(t)}\mathbf{a}_i + \lambda_i\mathbf{D}_\mathbf{A}^{(t)}\sum_{j=2}^{k+1}\alpha_{ij}\mathbf{a}_j. \tag{13}$$

By multiplying eigenvector $\mathbf{a}_p'$ $(2 \leq p \leq k+1, p \neq i)$ on both sides of Eq. (13) and taking advantage of the orthonormal property from Corollary 3.2, we obtain the following:

$$\begin{aligned} &\mathbf{a}_p'\Delta\mathbf{L}_\mathbf{A}\mathbf{a}_i + \mathbf{a}_p'\mathbf{D}_\mathbf{A}^{(t)}\sum_{j=2}^{k+1}\alpha_{ij}\lambda_j\mathbf{a}_j \\ &= \lambda_i\mathbf{a}_p'\Delta\mathbf{D}_\mathbf{A}\mathbf{a}_i + \Delta\lambda_i\mathbf{a}_p'\mathbf{D}_\mathbf{A}^{(t)}\mathbf{a}_i + \lambda_i\mathbf{a}_p'\mathbf{D}_\mathbf{A}^{(t)}\sum_{j=2}^{k+1}\alpha_{ij}\mathbf{a}_j \\ \Rightarrow\quad &\mathbf{a}_p'\Delta\mathbf{L}_\mathbf{A}\mathbf{a}_i + \alpha_{ip}\lambda_p = \lambda_i\mathbf{a}_p'\Delta\mathbf{D}_\mathbf{A}\mathbf{a}_i + \alpha_{ip}\lambda_i. \end{aligned} \tag{14}$$

Hence, the weight α_{ip} can be determined by:

$$\alpha_{ip} = \frac{\mathbf{a}_p'\Delta\mathbf{L}_\mathbf{A}\mathbf{a}_i - \lambda_i\mathbf{a}_p'\Delta\mathbf{D}_\mathbf{A}\mathbf{a}_i}{\lambda_i - \lambda_p}. \tag{15}$$

After eigenvector perturbation, we still need to make the orthonormal condition holds for new eigenvectors, thus we have $(\mathbf{a}_i+\Delta\mathbf{a}_i)'(\mathbf{D}_\mathbf{A}+\Delta\mathbf{D}_\mathbf{A})(\mathbf{a}_i+\Delta\mathbf{a}_i) = 1$. By expanding it and removing the second-order and third-order terms, we obtain the following equation:

$$2\mathbf{a}_i'\mathbf{D}_\mathbf{A}^{(t)}\Delta\mathbf{a}_i + \mathbf{a}_i'\Delta\mathbf{D}_\mathbf{A}^{(t)}\mathbf{a}_i = 0. \tag{16}$$

Then the solution of α_{ii} is as follows:

$$\alpha_{ii} = -\frac{1}{2}\mathbf{a}_i'\Delta\mathbf{D}_\mathbf{A}\mathbf{a}_i. \tag{17}$$

With the solutions of α_{ip} $(p \neq i)$ and α_{ii}, the perturbation of eigenvector $\mathbf{a}_i$ is given as follows:

$$\Delta\mathbf{a}_i = -\frac{1}{2}\mathbf{a}_i'\Delta\mathbf{D}_\mathbf{A}\mathbf{a}_i\mathbf{a}_i + \sum_{j=2,j\neq i}^{k+1}\left(\frac{\mathbf{a}_j'\Delta\mathbf{L}_\mathbf{A}\mathbf{a}_i - \lambda_i\mathbf{a}_j'\Delta\mathbf{D}_\mathbf{A}\mathbf{a}_i}{\lambda_i - \lambda_j}\right)\mathbf{a}_j. \tag{18}$$

Overall, the i-th eigen-pair $(\Delta\lambda_i, \Delta\mathbf{a}_i)$ can be updated on the fly by Eq. (12) and Eq. (18), the pseudocode of the updating process is illustrated in Algorithm 1. The first input is the top-k eigen-pairs of the generalized eigen-problem, they can be computed by standard methods like power iteration and Lanczos method [12]. Another input is the variation of the diagonal matrix and the Laplacian matrix. For the top-k eigen-pairs, we update eigenvalues in line 2 and update eigenvectors in line 3.

Likewise, the embedding of node attributes can also be updated in an online manner by Algorithm 1. Specifically, let $\mathbf{Y}_\mathbf{A}^{(t)}$ and $\mathbf{Y}_\mathbf{X}^{(t)}$ denote the embedding of network structure and node attributes at time step t, then at the following time step $t+1$, we first employ the proposed online model to update their embedding representations, then a final consensus embedding representation $\mathbf{Y}^{(t+1)}$ is derived by the correlation maximization method mentioned previously.

Algorithm 1 Updating of embedding results for the network

Input: Top-k eigen-pairs of the generalized eigen-problem $\{(\lambda_2, \mathbf{a}_2),(\lambda_3, \mathbf{a}_3),\ldots,(\lambda_{k+1}, \mathbf{a}_{k+1})\}$ at time t, variation of the diagonal matrix $\Delta\mathbf{L}_\mathbf{A}$ and Laplacian matrix $\Delta\mathbf{D}_\mathbf{A}$.

Output: Top-k eigen-pairs $\{(\lambda_2^{(t+1)}, \mathbf{a}_2^{(t+1)}),\ldots,(\lambda_{k+1}^{(t+1)}, \mathbf{a}_{k+1}^{(t+1)})\}$ at time step $t+1$.

1: **for** $i = 2$ to $k+1$ **do**
2: Calculate the variation of $\Delta\lambda_i$ by Eq. (12);
3: Calculate the variation of $\Delta\mathbf{a}_i$ by Eq. (18);
4: $\lambda_i^{(t+1)} = \lambda_i + \Delta\lambda_i$; $\mathbf{a}_i^{(t+1)} = \mathbf{a}_i + \Delta\mathbf{a}_i$;
5: **end for**

C - Computational Complexity Analysis. We theoretically analyze the computational complexity of the proposed online algorithm and show its superiority over the offline embedding methods.

Lemma 3.3. *The time complexity of the proposed online embedding algorithm over* T *time steps is* $O(Tk^2(n+l+l_a+l_x+d_x+d_x))$,

where k is the intermediate embedding dimension for network (or attributes), l is the final consensus embedding dimension, n is the number of nodes, and l_a, l_x, d_a, d_x are the number of non-zero entries in the sparse matrices $\Delta\mathbf{L_A}$, $\Delta\mathbf{L_X}$, $\Delta\mathbf{D_A}$, and $\Delta\mathbf{D_X}$, respectively.

Proof. In each time step, to update the top-k eigenvalues of the network and node attributes in an online fashion, it requires $O(k(d_a + l_a))$ and $O(k(d_x + l_x))$, respectively. Also, the online updating of the top-k eigenvectors for the network and attributes are $O(k^2(d_a+l_a+n))$ and $O(k^2(d_x+l_x+n))$, respectively. After that, the complexity for the consensus embedding is $O(k^2l)$. Therefore, the computational complexity of the proposed online model over T time steps are $O(Tk^2(n + l + l_a + l_x + d_x + d_x))$. □

Lemma 3.4. *The time complexity of the proposed offline embedding algorithm over T time steps is $O(Tn^2(k + l))$, where k is the intermediate embedding dimension for network (or attributes), l is the final consensus embedding dimension.*

Proof. Omitted for brevity. □

As can be shown, since $\Delta\mathbf{L_A}$, $\Delta\mathbf{L_X}$, $\Delta\mathbf{D_A}$, and $\Delta\mathbf{D_X}$ are often very sparse, thus l_a, l_x, d_x, d_a are usually very small, meanwhile we have $k \ll n$ and $l \ll n$. Based on the above analysis, the proposed online embedding algorithm for dynamic attributed networks is much more efficient than rerunning the offline method repeatedly.

4 EXPERIMENTS

In this section, we conduct experiments to evaluate the effectiveness and efficiency of the proposed DANE framework for dynamic attributed network embedding. In particular, we attempt to answer the following two questions: (1) *Effectiveness*: how effective are the embeddings obtained by DANE on different learning tasks? (2) *Efficiency*: how fast is the proposed framework DANE compared with other offline embedding methods? We first introduce the datasets and experimental settings before presenting details of the experimental results.

4.1 Datasets

We use four datasets BlogCatalog, Flickr, Epinions and DBLP for experimental evaluation. Among them, BlogCatalog and Flickr are synthetic data from static attributed networks, and they have been used in previous research [19, 20]. We randomly add 0.1% new edges and change 0.1% attribute values at each time step to simulate its evolving nature. The other two datasets, Epinions and DBLP are real-world dynamic attributed networks. Epinions is a product review site in which users share their reviews and opinions about products. Users themselves can also build trust networks to seek advice from others. Node attributes are formed by the bag-of-words model on the reviews, while the major categories of reviews by users are taken as the ground truth of class labels. The data has 16 different time steps. In the last dataset DBLP, we extracted a DBLP co-author network for the authors that publish at least two papers between the years of 2001 and 2016 from seven different areas. Bag-of-words model is applied on the paper title to obtain the attribute information, and the major area the authors publish is considered as ground truth. The detailed statistics of these datasets are listed in Table 2.

	BlogCatalog	Flickr	Epinions	DBLP
# Nodes	5,196	7,575	14,180	23,393
# Attributes	8,189	12,047	9,936	8,945
# Edges	173,468	242,146	227,642	289,478
# Classes	6	9	20	7
# Time Steps	10	10	16	16

Table 2: Detailed information of the datasets.

4.2 Experimental Settings

One commonly adopted way to evaluate the quality of the embedding representation [7, 17, 26, 33] is by the following two unsupervised and supervised tasks: network clustering and node classification. First, we validate the effectiveness of the embedding representations by DANE on the network clustering task. Two standard clustering performance metrics, i.e., *clustering accuracy* (ACC) and *normalized mutual information* (NMI) are used. In particular, after obtaining the embedding representation of each node in the attributed network, we perform K-means clustering based on the embedding representations. The K-means algorithm is repeated 10 times and the average results are reported since K-means may converge to the local minima due to different initializations. Another way to assess the embedding is by the node classification task. Specifically, we split the the embedding representations of all nodes via a 10-fold cross-validation, using 90% of nodes to train a classification model by logistic regression and the rest 10% nodes for the testing. The whole process is repeated 10 times and the average performance are reported. Three evaluation metrics, *classification accuracy*, *F1-Macro* and *F1-Micro* are used. How to determine the optimal number of embedding dimensions is still an open research problem, thus we vary the embedding dimension as $\{10, 20, ..., 100\}$ and the best results are reported.

4.2.1 Baseline Methods. DANE is measured against the following baseline methods on the two aforementioned tasks:

- **Deepwalk**: learns network embeddings by word2vec and truncated random walk techniques [26].
- **LINE**: learns embeddings by preserving the first-order and second-order proximity structures of the network [33].
- **DANE-N**: is a variation of the proposed DANE with only network information.
- **DANE-A**: is a variation of the proposed DANE with only attribute information.
- **CCA**: directly uses the original network structure and attributes for a joint low-dimensional representation [14].
- **LCMF**: maps network and attributes to a shared latent space by collective matrix factorization [44].
- **LANE**: is a label informed attributed network embedding method, we use one of its variant LANE w/o Label [16].
- **DANE-O**: is a variation of DANE that reruns the offline model at each time step.

It is important to note that Deepwalk, LINE, CCA, LCMF, LANE, and DANE-O can only handle static networks. To have a fair comparison with the proposed DANE framework, we rerun these baseline methods at each time step and report the average performance

over all time steps[1]. We follow the suggestions of the original papers to set the parameters of all these baselines.

4.3 Unsupervised Task - Network Clustering

To evaluate the effectiveness of embedding representations, we first compare DANE with baseline methods on network clustering which is naturally an unsupervised learning task. As per the fact that the attributed networks are constantly evolving, we compare the average clustering performance over all time steps. The average clustering performance comparison w.r.t. ACC and NMI are presented in Table 3. We make the following observations:

- DANE and its offline version DANE-O consistently outperform all baseline methods on four dynamic attributed networks by achieving better clustering performance. We also perform pairwise Wilcoxon signed-rank test [11] between DANE, DANE-O and these baseline methods and the test results show that DANE and DANE-O are significantly better (with both 0.01 and 0.05 significance levels).
- DANE, DANE-O and LANE achieve better clustering performance than network embedding methods such as Deepwalk, LINE and DANE-N and attribute embedding method DANE-A. The improvements indicate that attribute information is complementary to pure network topology and can help learn more informative embedding representations. Meanwhile, DANE also outperforms the CCA and LCMF which also leverage node attributes. The reason is that although these methods learn a low-dimensional representation by using both sources, they are not explicitly designed to preserve the node proximity. Also, their performance degenerates when the data is very noisy.
- Even though DANE leverages matrix perturbation theory to update the embedding representations, its performance is very close to DANE-O which reruns at each time step. It implies that the online embedding model does not sacrifice too much informative information in terms of embedding.

4.4 Supervised Task - Node Classification

Next, we assess the effectiveness of embedding representations on a supervised learning task - node classification. Similar to the settings of network clustering, we report the average classification performance over all time steps. The classification results in terms of three different measures are shown in Table 4. The following findings can be inferred from the table:

- Generally, we have the similar observations as the clustering task. The methods which only use link information or node attributes (e.g., Deepwalk, LINE, DANE-N, DANE-A) and methods which do not explicitly model node proximity (e.g., CCA, LCMF) give poor classification results.
- The embeddings learned by DANE and DANE-O help train a more discriminative classification model by obtaining higher classification performance. In addition, pairwise Wilcoxon signed-rank test [11] shows that DANE and DANE-O are significantly better.
- For the node classification task, the attribute embedding method DANE-A works better than the network embedding method in the BlogCatalog, Flickr and Epinions datasets. The reason is that in these datasets, the class labels are more closely related to the attribute information than the network structure. However, it is a different case for the DBLP dataset in which the labels of authors are more closely related to the coauthor relationships.

[1]For baseline methods that cannot finish in 24hrs, we only run it once. As networks evolve smoothly, there is not much difference in terms of average performance.

4.5 Efficiency of Online Embedding

To evaluate the efficiency of the proposed DANE framework, we compare DANE with several baseline methods CCA, LCMF, LANE which also use two data representations. Also, we include the offline version of DANE, i.e., DANE-O. As all these methods are not designed to handle network dynamics, we compare their cumulative running time over all time steps and plot it in a log scale. As can be observed from Figure 2, the proposed DANE is much faster than all these comparison methods. In all these datasets, it terminates within one hour while some offline methods need several hours or even days to run. It can also be shown that both DANE and DANE-O are much faster than all other offline methods. To be more specific, for example, DANE is 84×, 21× and 14× faster than LCMF, CCA and LANE respectively on Flickr dataset. To further investigate the superiority of DANE against its offline version DANE-O, we compare the speedup rate of DANE against DANE-O w.r.t. different embedding dimensions in Figure 3. As can be observed, when the embedding dimension is small (around 10), DANE achieves around 8×, 10×, 8×, 12× speedup on BlogCatalog, Flickr, Epinions, and DBLP, respectively. When the embedding dimensionality gradually increases, the speedup of DANE decreases, but it is still significantly faster than DANE-O. With all the above observations, we can draw a conclusion that the proposed DANE framework is able to learn informative embeddings for attributed networks efficiently without jeopardizing the classification and the clustering performance.

5 RELATED WORK

We briefly review related work from (1) network embedding; (2) attributed network mining; and (3) dynamic network analysis.

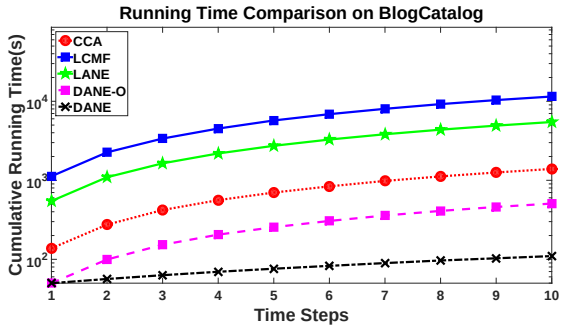
The pioneer of network embedding can be dated back to the 2000s when many graph embedding algorithms [4, 28, 35] were proposed. These methods target to build an affinity matrix that preserves the local geometry structure of the data manifold and then embed the data to a low-dimensional representation. Motivated by the graph embedding techniques, Chen et al. [9] proposed one of the first network embedding algorithms for directed networks. They used random walk to measure the proximity structure of the directed network. Recently, network embedding techniques have received a surge of research interests in network science. Among them, Deepwalk [26] generalizes the word embedding and employs a truncated random walk to learn latent representations of a network. Node2vec [13] further extend Deepwalk by adding the flexibility in exploring node neighborhoods. LINE [33] carefully designs an optimized objective function that preserves first-order and second-order proximities to learn network representations. GraRep [6] can be regarded as an extension of

Table 3: Clustering results (%) comparison of different embedding methods.

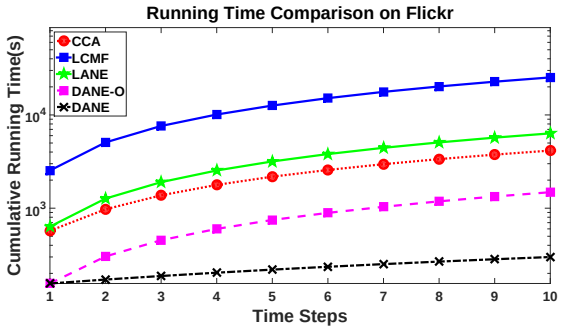
Datasets		BlogCatalog		Flickr		Epinions		DBLP	
Methods		ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
Network	Deepwalk	49.85	30.51	40.70	24.29	13.31	12.72	53.61	32.54
	LINE	50.20	29.53	42.93	26.01	14.34	12.65	51.61	30.74
	DANE-N	37.05	21.84	31.89	18.91	12.01	11.95	56.61	31.54
Attributes	DANE-A	62.32	45.95	63.80	48.29	16.12	11.62	47.37	20.64
Network+Attributes	CCA	33.42	11.86	24.39	10.89	10.85	8.61	26.42	18.60
	LCMF	55.72	40.38	27.03	13.06	12.86	10.73	42.27	26.48
	LANE	65.06	48.89	65.45	52.58	32.18	22.09	55.80	31.84
	DANE-O	**80.31**	**59.46**	**67.33**	**53.04**	**34.11**	**23.07**	**59.14**	**35.31**
	DANE	**79.69**	**59.32**	**67.24**	**52.19**	**34.52**	**22.36**	**57.68**	**34.87**

Table 4: Classification results (%) comparison of different embedding methods.

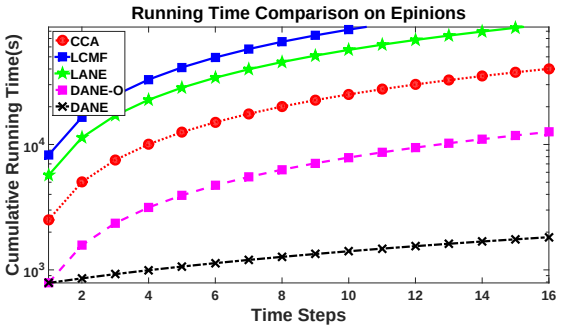
Datasets		BlogCatalog			Flickr			Epinions			DBLP		
Methods		AC	Micro	Macro	AC	Micro	Macro	AC	Micro	Macro	AC	Micro	Macro
Network	Deepwalk	68.05	67.15	68.18	60.08	58.93	59.08	22.12	17.43	20.10	74.38	69.65	72.37
	LINE	70.20	69.88	70.91	61.03	60.90	60.01	23.54	17.17	21.05	72.97	67.56	70.97
	DANE-N	66.97	66.06	67.78	49.37	47.82	49.34	21.25	20.57	21.88	71.99	65.33	71.94
Attributes	DANE-A	80.23	79.86	80.23	76.66	75.59	76.60	23.76	21.57	22.00	63.92	54.80	62.97
Network+Attributes	CCA	48.63	49.96	49.63	27.09	26.54	26.09	11.53	9.43	10.56	45.67	42.08	43.83
	LCMF	84.41	89.01	**89.26**	66.27	66.75	65.71	19.14	9.22	10.14	69.71	68.01	68.42
	LANE	87.52	87.52	87.93	77.54	77.81	77.26	27.74	28.45	28.87	72.15	71.09	73.48
	DANE-O	**89.34**	**89.15**	89.23	**79.68**	**79.52**	**79.95**	**31.23**	**31.28**	**31.35**	**77.21**	**74.96**	**75.48**
	DANE	**89.09**	**88.78**	88.94	**79.56**	**78.94**	**79.56**	**30.87**	**30.93**	**30.81**	**76.64**	**74.53**	**75.69**



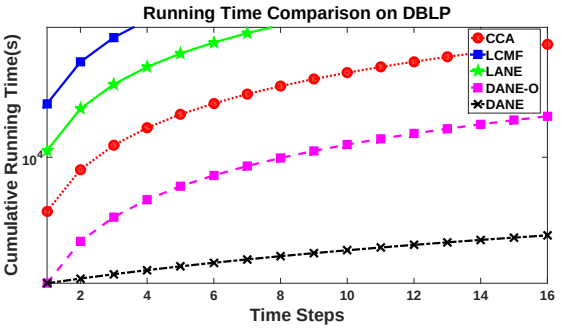
(a) BlogCatalog



(b) Flickr

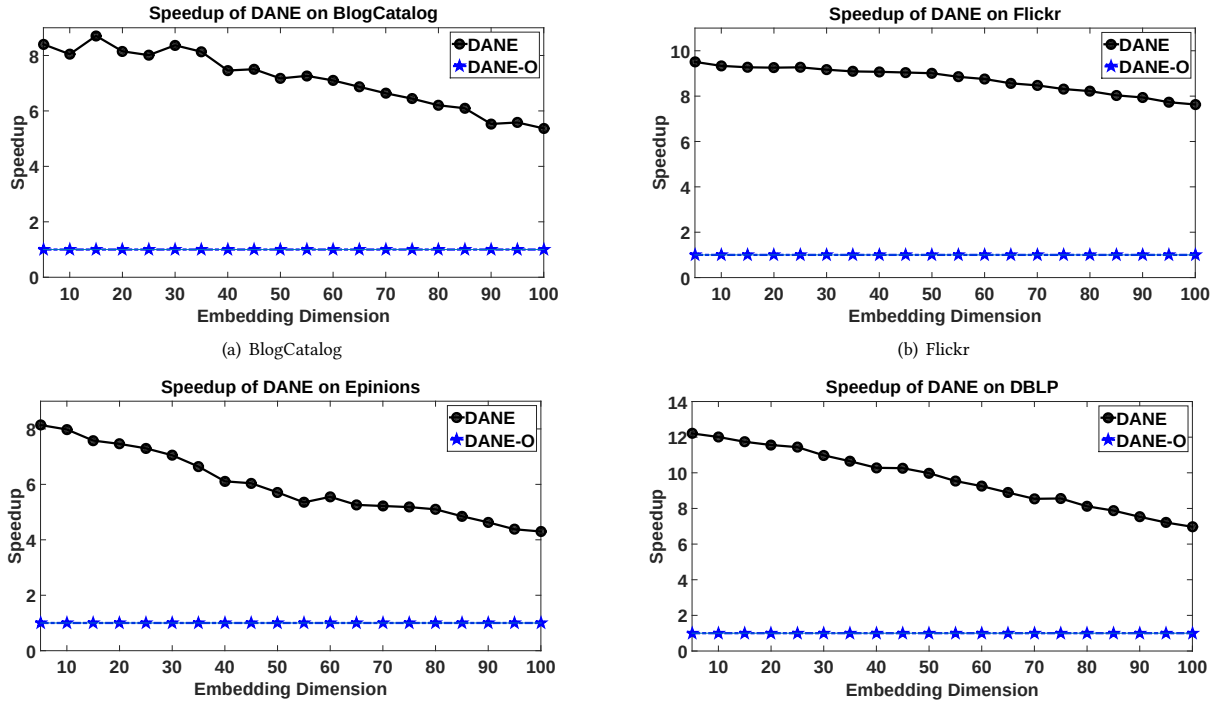


(c) Epinions



(d) DBLP

Figure 2: Cumulative running time comparison.



(a) BlogCatalog (b) Flickr

(c) Epinions (d) DBLP

Figure 3: Running time speedup of DANE against its offline version DANE-O.

LINE which considers high-order information. Most recently, some deep learning based approaches are proposed to enhance the learned embeddings [38, 42].

All the above mentioned approaches, however, are limited to deal with plain networks. In many cases, we are often faced with attributed networks. Many efforts have been devoted to gain insights from attributed networks. For example, Zhu et al. [44] proposed a collective matrix factorization model that learns a low-dimensional latent space by both the links and node attributes. Similar matrix factorization based methods are proposed in [40, 43]. Chang et al. [7] used deep learning techniques to learn a joint feature representation for heterogeneous networks. Huang et al. [16] studied whether label information can help learn better feature representation in attributed networks. Instead of directly learning embeddings, another way is to perform unsupervised feature selection [20, 32]. Nevertheless, all these methods can only handle static networks; it is still not clear how to learn embedding representations efficiently when attributed networks are constantly evolving over time.

As mentioned above, many real-world networks, especially social networks, are not static but are continuously evolving. Hence, the results of many network mining tasks will become stale and need to be updated to keep freshness. For example, Tong et al. [36] proposed an efficient way to sample columns and/or rows from the network adjacency matrix to achieve low-rank approximation. In [34], the authors employed the temporal information to analyze the multi-mode network when multiple interactions are evolving. Ning et al. [24] proposed an incremental approach to perform spectral clustering on networks dynamically. Aggarwal and Li [3] proposed a random-walk based method to perform dynamic classification in content-based networks. In [8], a fast eigen-tracking algorithm is proposed which is essential for many graph mining algorithms involving adjacency matrix. Li et al. [18] studied how to perform unsupervised feature selection in a dynamic and connected environment. A more detailed review of dynamic network analysis can be referred to [2]. However, all these methods are distinct from our proposed framework as we are the first to tackle the problem of attributed network embedding in a dynamic environment.

6 CONCLUSIONS AND FUTURE WORK

The prevalence of attributed networks in many real-world applications presents new challenges for many learning problems because of its natural heterogeneity. In such networks, interactions among networked instances tend to evolve gradually, and the associated attributes also change accordingly. In this paper, we study a novel problem: how to learn embedding representations for nodes in dynamic attributed networks to enable further learning tasks. In particular, we first build an offline model for a consensus embedding presentation which could capture node proximity in terms of both network topology and node attributes. Then in order to capture the evolving nature of attributed network, we present an efficient online method to update the embeddings on the fly. Experimental results on synthetic and real dynamic attributed networks demonstrate the efficacy and efficiency of the proposed framework.

There are many future research directions. First, in this paper, we employ first-order matrix perturbation theory to update the

embedding representations in an online fashion. We would like to investigate how the high-order approximations can be applied to the online embedding learning problem. Second, this paper focuses on online embedding for two different data representations; we also plan to extend the current framework to multi-mode and multi-dimensional dynamic networks.

REFERENCES

[1] Lada A Adamic and Bernardo A Huberman. 2000. Power-law distribution of the world wide web. *Science* 287, 5461 (2000), 2115–2115.

[2] Charu Aggarwal and Karthik Subbian. 2014. Evolutionary network analysis: A survey. *Comput. Surveys* 47, 1 (2014), 10.

[3] Charu C Aggarwal and Nan Li. 2011. On node classification in dynamic content-based networks. In *SDM*. SIAM, 355–366.

[4] Mikhail Belkin and Partha Niyogi. 2001. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering.. In *NIPS*. 585–591.

[5] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. 2011. Node classification in social networks. In *Social network data analytics*. 115–148.

[6] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *CIKM*. ACM, 891–900.

[7] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. 2015. Heterogeneous network embedding via deep architectures. In *KDD*. 119–128.

[8] Chen Chen and Hanghang Tong. 2015. Fast Eigen-Functions Tracking on Dynamic Graphs. In *SDM*. 559–567.

[9] Mo Chen, Qiong Yang, and Xiaoou Tang. 2007. Directed Graph Embedding.. In *IJCAI*. 2707–2712.

[10] Yun Chi, Xiaodan Song, Dengyong Zhou, Koji Hino, and Belle L Tseng. 2007. Evolutionary spectral clustering by incorporating temporal smoothness. In *KDD*. 153–162.

[11] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *JMLR* 7 (2006), 1–30.

[12] Gene H Golub and Charles F Van Loan. 2012. *Matrix computations*. Vol. 3.

[13] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. ACM, 855–864.

[14] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural computation* 16, 12 (2004), 2639–2664.

[15] Xia Hu, Jiliang Tang, Yanchao Zhang, and Huan Liu. 2013. Social Spammer Detection in Microblogging.. In *IJCAI*. 2633–2639.

[16] Xiao Huang, Jundong Li, and Xia Hu. 2017. Label Informed Attributed Network Embedding. In *WSDM*. ACM, 731–739.

[17] Yann Jacob, Ludovic Denoyer, and Patrick Gallinari. 2014. Learning latent representations of nodes for classifying in heterogeneous social networks. In *WSDM*. 373–382.

[18] Jundong Li, Xia Hu, Ling Jian, and Huan Liu. 2016. Toward Time-Evolving Feature Selection on Dynamic Networks. In *ICDM*. IEEE, 1003–1008.

[19] Jundong Li, Xia Hu, Jiliang Tang, and Huan Liu. 2015. Unsupervised Streaming Feature Selection in Social Media. In *CIKM*. 1041–1050.

[20] Jundong Li, Xia Hu, Liang Wu, and Huan Liu. 2016. Robust Unsupervised Feature Selection on Networked Data. In *SDM*.

[21] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *JASIST* 58, 7 (2007), 1019–1031.

[22] Peter V Marsden and Noah E Friedkin. 1993. Network studies of social influence. *Sociological Methods & Research* 22, 1 (1993), 127–151.

[23] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* (2001), 415–444.

[24] Huazhong Ning, Wei Xu, Yun Chi, Yihong Gong, and Thomas S Huang. 2007. Incremental Spectral Clustering With Application to Monitoring of Evolving Blog Communities.. In *SDM*. 261–272.

[25] Beresford N Parlett. 1980. *The symmetric eigenvalue problem*. Vol. 7.

[26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.

[27] Joseph J Pfeiffer III, Sebastian Moreno, Timothy La Fond, Jennifer Neville, and Brian Gallagher. 2014. Attributed graph models: Modeling network structure with correlated attributes. In *WWW*. ACM, 831–842.

[28] Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science* (2000).

[29] Cosma Rohilla Shalizi and Andrew C Thomas. 2011. Homophily and contagion are generically confounded in observational social network studies. *Sociological methods & research* 40, 2 (2011), 211–239.

[30] Gilbert W Stewart. 1990. Matrix perturbation theory. (1990).

[31] Chenhao Tan, Lillian Lee, Jie Tang, Long Jiang, Ming Zhou, and Ping Li. 2011. User-level sentiment analysis incorporating social networks. In *KDD*. ACM, 1397–1405.

[32] Jiliang Tang and Huan Liu. 2012. Unsupervised feature selection for linked social media data. In *KDD*.

[33] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *WWW*. 1067–1077.

[34] Lei Tang, Huan Liu, Jianping Zhang, and Zohreh Nazeri. 2008. Community evolution in dynamic multi-mode networks. In *KDD*. 677–685.

[35] Joshua B Tenenbaum, Vin De Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290, 5500 (2000), 2319–2323.

[36] Hanghang Tong, Spiros Papadimitriou, Jimeng Sun, Philip S Yu, and Christos Faloutsos. 2008. Colibri: fast mining of large static and dynamic graphs. In *KDD*. 686–694.

[37] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.

[38] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *KDD*. ACM, 1225–1234.

[39] Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Albert-Laszlo Barabasi. 2011. Human mobility, social ties, and link prediction. In *KDD*. 1100–1108.

[40] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network Representation Learning with Rich Text Information. In *IJCAI*. 2111–2117.

[41] Tianbao Yang, Rong Jin, Yun Chi, and Shenghuo Zhu. 2009. Combining link and content for community detection: a discriminative approach. In *KDD*. 927–936.

[42] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *ICML*. 40–48.

[43] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2016. Collective Classification via Discriminative Matrix Factorization on Sparsely Labeled Networks. In *CIKM*. ACM, 1563–1572.

[44] Shenghuo Zhu, Kai Yu, Yun Chi, and Yihong Gong. 2007. Combining content and link for classification using matrix factorization. In *SIGIR*. 487–494.