Received March 24, 2018, accepted May 8, 2018, date of publication May 25, 2018, date of current version June 20, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2839770

Deep Dynamic Network Embedding for Link Prediction

TAISONG LI[1,2], JIAWEI ZHANG[3], (Member, IEEE), PHILIP S. YU[4], (Fellow, IEEE), YAN ZHANG[1], AND YONGHONG YAN[1,2,5]

[1]Key Laboratory of Speech Acoustics and Content Understanding, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China
[2]Department of Physics, University of Chinese Academy of Sciences, Beijing 101408, China
[3]Department of Computer Science, Florida State University, Tallahassee, FL 32304, USA
[4]Department of Computer Science, The University of Illinois at Chicago, Chicago, IL 60607, USA
[5]Xinjiang Key Laboratory of Minority Speech and Language Information Processing, Xinjiang Technical Institute of Physics and Chemistry, Chinese Academy of Sciences, Ürümqi 830011, China

Corresponding author: Taisong Li (tythonlee@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grants 11590770-4, 61650202, 11722437, U1536117, 61671442, 11674352, 11504406, and 61601453, in part by the National Key Research and Development Program under Grants 2016YFB0801203, 2016YFC0800503, and 2017YFB1002803, and in part by the Key Science and Technology Project of the Xinjiang Uygur Autonomous Region under Grant 2016A03007-1.

ABSTRACT Network embedding task aims at learning low-dimension latent representations of vertices while preserving the structure of a network simultaneously. Most existing network embedding methods mainly focus on static networks, which extract and condense the network information without temporal information. However, in the real world, networks keep evolving, where the linkage states between the same vertex pairs at consequential timestamps have very close correlations. In this paper, we propose to study the network embedding problem and focus on modeling the linkage evolution in the dynamic network setting. To address this problem, we propose a deep dynamic network embedding method. More specifically, the method utilizes the historical information obtained from the network snapshots at past timestamps to learn latent representations of the future network. In the proposed embedding method, the objective function is carefully designed to incorporate both the network internal and network dynamic transition structures. Extensive empirical experiments prove the effectiveness of the proposed model on various categories of real-world networks, including a human contact network, a bibliographic network, and e-mail networks. Furthermore, the experimental results also demonstrate the significant advantages of the method compared with both the state-of-the-art embedding techniques and several existing baseline methods.

INDEX TERMS Social network analysis, network embedding, link prediction, deep learning.

I. INTRODUCTION

Network structured data can effectively model various types of linked data in the real world, in which nodes represent entities and edges indicate connections. Mining information from network is an important problem and is ubiquitous in real-world applications. For example, the recommendation system in Youtube or Amazon aims to predict the potential videos/products users can be interested in, which can be modeled as a user-item link prediction problem. The key point for these applications is how to learn useful information from network structures. One of the most effective representation learning approaches for networked data is network embedding, which aims to map the network into a low-dimensional space. Such a network embedding method is proved to be very effective in link prediction or classification tasks. It is generally applied to static networks such as bibliographic network, email network and online social network, etc.

As introduced in [1], few networks in the real-world are actually static but keep evolving with time. For instance, bibliographical network [2], online social network [3] and email network [4] change non-linearly from each snapshots. Representation learning for dynamic network is not an easy task, which needs to model both the network structure and temporal information properly to preserve network information.

In the past decades, many network embedding methods have been proposed. Most of these methods employ shallow models, such as IsoMAP [5], Laplacian Eigenmaps (LE) [6] and LINE [7]. They can efficiently extract information from

a network since the shallow model structure has less cost in computation. However, the underlying network structure from high dimension to low-dimensional space is highly non-linear [8], which cannot be effectively captured by these methods because of their limited representation abilities. To solve this problem, some of the deep models such as SDNE [9], node2vec [10] and DeepWalk [11] have been proposed. Due to the deep neural network structure, these methods have a high ability to model the non-linear transformation of network structure. They can preserve linkage information for each node as well. However, these works have thus far focused on representation learning for static networks, in which they only have singular snapshot of nodes and relationships. To the best of our knowledge, dynamic network embedding is still an open problem to this context so far. Some network evolving methods like tRBM [12], ctRBM [3] are competent to capture the evolution pattern. However, they have limited ability to predict links due to their shallow learning process.

Learning embedding from dynamic networks faces the following great challenges:

- Incorporating structure information: most embedding approaches try to preserve information of network by using its current structure. How to incorporate historical and current dataset to learn future representation is a great challenge for network embedding.
- Highly non-linear transformations: evolving non-linearly over time is commonly seen in dynamic networks with periodic fluctuations. How to catch these non-linearities in dynamic linkage changing patterns for network representation learning is a difficult problem.
- Node interactions: a network can be presented as an adjacency matrix in each time slice, where the node local neighborhood structure can be effectively captured by the row vectors corresponding to the nodes in the adjacency matrix. Generally, most existing network embedding methods take the matrix as an input on the premise that vectors are independent to each other. However, interactions between nodes also contain structure information. How to model the correlation between nodes along with the change of network is an important factor for structure preservation.

In order to address structure information incorporation problem, we propose to use historical linkage status and current network structure to model network evolving patterns. Then, employing the trained model to infer the future network structure. This is motivated by the recent success of dynamic network learning method, which has been demonstrated to have a powerful inference ability for link prediction [3]. In particular, our proposed model deploys previous structures to learn the node presentations of future networks. After that, the node representations can be used to infer the next linkage status of network.

In order to capture the non-linear transformations from historical snapshots, we propose a new deep model to learn vertex representations for dynamic networks. This is motivated by the temporal deep learning model Recurrent Neural Network (RNN), which has achieved substantial success in modeling sequential data in various disciplines, e.g., natural language processing and speech recognition. We design a multi-layer architecture which consists of the encoder and decoder layers respectively. The encoder layer accepts sequence data input and learns the latent representation through multiple non-linear functions successively. Then, the output is fed to the up-layer for decoding. Since there are various non-linear functions in encoder and decoder layers, we can map the historical data into highly non-linear latent space. As a result, the proposed deep model is able to learn the complicated transformations of each vertex.

To preserve the information of node interactions, we further propose to exploit interaction proximity in the learning process. As is known that nodes contacted in the history tend to connect in the future network. The interaction proximity is designed to measure such contact closeness between two nodes. Technically, traditional RNN-based deep models take the input samples independently to each other. It is capable to capture the transition pattern for each node, but ignores the correlations between node vectors. Thus, the original deep model may fail to utilize the abundant node correlation information to model the evolving patterns and infer network structure. To resolve such a shortcoming, a novel network interaction proximity term is introduced in this paper, which measures the correlations between nodes. The network interaction proximity term greatly enriches the proposed deep dynamic network embedding model, and make it possible to capture network internal connection structure.

To demonstrate our model's potential in real world scenarios, we conduct experiments on various categories of real-world networks and evaluate its performance on link prediction task specifically. All these used network data are dynamic in a certain time period. The result shows that compared with the state-of-the-art and several existing baseline methods, the proposed method can infer the networks to be prominently better and achieve substantial gains on various networks. Such a result demonstrates that our algorithm has the capacity to capture the network dynamics the link prediction task.

Our contributions are summarized as follows:

(1) We propose a deep learning model to perform embedding on dynamic networks. The method is able to capture the non-linear transformations of nodes and preserve dynamic networks' structure. To the best of our knowledge, we are among the first to learn dynamic network representations.

(2) The proposed new deep architecture, which can be employed as a generative model, to infer dynamic network embedding by using historical snapshots. It demonstrates a better performance on link prediction than traditional embedding methods and generative models.

(3) To optimize our deep model, we further introduce the interaction proximity concept, which preserves the information of node interactions along an evolving period. The results indicate that the deep model achieves

substantial gains when interaction proximity has been considered.

The rest of the paper is arranged as follows. We will first define several concepts in Section II. Section III discusses our proposed dynamic network embedding method for link prediction. Section IV describes the experimental results, while the conclusions are presented in Section V.

II. PROBLEM DEFINITION

We first generally declare the notations used in this paper, then we formally define several important terminologies and introduce the formulation of dynamic network embedding problem.

A. NOTATIONS

In the sequel of this paper, we will use the lower case letters (e.g., x) to represent scalars, lower case bold letters (e.g., $\mathbf{x}$) to denote column vectors, bold-face upper case letters (e.g., $\mathbf{X}$) to denote matrices, and upper case calligraphic letters (e.g., $\mathcal{X}$) to denote sets. Given a matrix $\mathbf{X}$, we denote $\mathbf{X}(i,:)$ and $\mathbf{X}(:,j)$ as the i_{th} row and j_{th} column of matrix $\mathbf{X}$ respectively. The (i_{th}, j_{th}) entry of matrix $\mathbf{X}$ can be denoted as either $X(i,j)$ or $X_{i,j}$, which will be used interchangeably in this paper. We use $\mathbf{X}^\top$ and $\mathbf{x}^\top$ to represent the transpose of matrix $\mathbf{X}$ and vector $\mathbf{x}$. For vector $\mathbf{x}$, we represent its L_p-norm as $\|\mathbf{x}\|_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$. The Frobenius norm of matrix $\mathbf{X}$ can be represented as $\|\mathbf{X}\|_F = (\sum_{i,j} |X_{i,j}|^2)^{\frac{1}{2}}$. The element-wise product of vectors $\mathbf{x}$ and $\mathbf{y}$ of the same dimension is represented as $\mathbf{x} \odot \mathbf{y}$, while the element-wise product of matrices $\mathbf{X}$ and $\mathbf{Y}$ of the same dimensions is represented as $\mathbf{X} \odot \mathbf{Y}$.

B. DEFINITIONS

Definition 1 (Network): A network can be represented by a graph: $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ refers to a set of nodes, and $\mathcal{E} \subseteq |\mathcal{V}| \times |\mathcal{V}|$ represents a set of links among the nodes. Each edge $e \subseteq \mathcal{E}$ is an unordered pair $e = (v_i, v_j)$ and is associated with a weight w_{ij}, which indicates the strength of the relation. For unweighted graph $w_{ij} = 1$ and for weighted graph, $w_{ij} > 0$.

Considering the input of our deep model is a matrix, we denote the adjacency matrix as $\mathbf{X} \in \mathbb{R}^{n \times n}$. Each row of the matrix indicates a user's link vector, which can be presented as $\mathbf{X}(i,:)$. Each element of $\mathbf{X}$ is written as X_{ij}, which means the link state between node i and node j.

Definition 2 (Dynamic Network): In dynamic networks, we denote a series of snapshots as $\{\mathcal{G}_{t-N}, \ldots, \mathcal{G}_{t-1}, \mathcal{G}_t\}$, which represent the state of the network at each time slice (N denotes the target time window size). We follow the dynamic network settings in [3] that the nodes $\mathcal{V}$ remain constant while the edges $\mathcal{E}_t$ change when network evolving. Hence, we can represent graph $\mathcal{G}$ at each time t as $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t)$.

The adjacency matrix in dynamic network is similar to the static network except the timestamps. For each time slice t, the adjacency matrix is presented as $\mathbf{X}^t$. The element is written as X_{ij}^t, which means the link state between node i and node j at time t.

In practice, a temporal deep model, like RNN, can be used for sequence data and model its pattern of changes. However, it treats each sample independently so that vertexes' interaction over time can not be captured for a dynamic network. Thus, we further proposed an interaction proximity that can optimize the temporal deep model by taking samples' correlation into account.

Definition 3 (Interaction Proximity): The interaction proximity describes the closeness between vertexes. For v_i and v_j, if they have any linkage at any snapshot, there exists positive interaction proximity. If no edge is observed, their interaction proximity is 0.

Intuitively, people are more likely to contact with someone who has been acquainted before. Such a phenomenon has been observed in many fields. For example, in co-authorship network, researchers tend to cooperate with others who have published papers together. In mail network, most e-mails are sent to the people you have contacted before. The interaction proximity considers connections through all snapshots, then it use this contact frequency to define the acquaintance between users. Therefore, it can highly enrich the relationship of vertexes, and it is able to preserve contact information and alleviate network embedding problem.

With a deep model and interaction proximity, we investigate the problem of how to integrate them simultaneously to model the structure evolving for dynamic network representation learning. Formally, the problem is defined as follows:

Definition 4 (Deep Dynamic Network Embedding): Given a network with temporal information $\mathcal{G}$, it can be sliced into a series snapshots as $\{\mathcal{G}_{t-N}, \ldots, \mathcal{G}_{t-1}, \mathcal{G}_t\}$. The dynamic network embedding problem aims to learn the low-dimensional latent representations $\mathbf{X}^t \in \mathbb{R}^{n \times d}$ by using historical networks $\mathcal{G}_h = \{\mathcal{G}_{t-N}, \ldots, \mathcal{G}_{t-1}\}$. The latent embeddings with dimension $d \ll |\mathcal{V}|$ are able to capture and recover the network structure at time t. A general learning and inference process is illustrated in Figure 1.

III. DDNE: DEEP DYNAMIC NETWORK EMBEDDING

In this section, we present a general framework, DDNE, which is capable of learning desirable node representations in dynamic networks. This framework is inspired by RNNsearch [13], which is proposed to cope with machine translation problem. Most of the neural machine translation models belong to a family of encoder-decoders [14], [15], which takes each sample independently for both encoder and decoder. However, our proposed method is well reshaped and the correlations between samples are fully considered to preserve information of networks. In the subsection, we first introduce the gated recurrent unit. Then, we explicitly describe the proposed new architecture of deep dynamic embedding method. After that, we introduce the loss functions and the optimization of the algorithm. At last, we further discuss the training and inference of DDNE model.

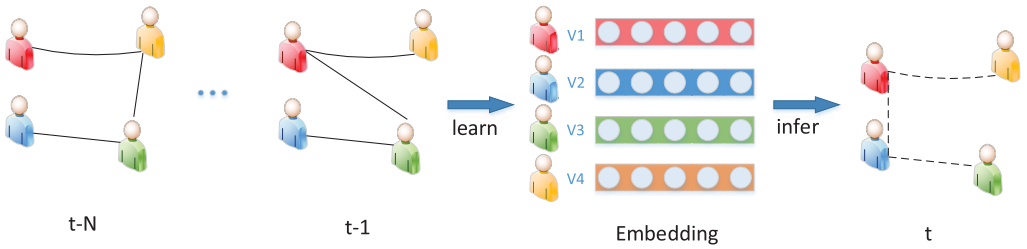


FIGURE 1. Learning and inference process of link prediction.

TABLE 1. Terms and notations.

Symbol	Definition
n	number of nodes
N	number of time slices
$\mathbf{X}(i,:) = \{\mathbf{X}^k(i,:)\}_{k=t}^{t-N}$	the adjacency vectors of node i
$\hat{\mathbf{X}}$	the output of decoder
$\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}, \mathbf{U}_z, \mathbf{U}_r, \mathbf{U}$	the weight matrices of GRU
$\mathbf{W}^{(m)}$	the m-th layer weight of decoder
$\mathbf{b}^{(m)}$	the m-th layer bias of decoder
$\mathbf{c}_i$	the output of encoder for node i

Before introducing the deep dynamic embedding model, we define some of the terms and notations in Table 1 which will be used later. Note that the adjacency vector contains input and target vectors, which is divided by different timestamps.

A. PRELIMINARY: GATED RECURRENT UNIT

A gated recurrent unit (GRU) was proposed by Cho *et al.* [17] to make each recurrent unit to adaptively capture dependencies of different time scales. Empirical experiments on sequential datasets demonstrated that the RNNs with the gating units (GRU-RNN and LSTM-RNN) clearly outperformed the traditional tanh-RNN in terms of prediction accuracy or convergence speed. Furthermore, the performance of GRU and LSTM is so comparable but the GRU generally makes faster progress than LSTM in terms of both the number of updates and actual CPU time [18]. Thus, we take the GRU as encoder units to modeling the dynamic network data.

The GRU structure is illustrated in Figure 2, and the functions are defined as follows:

$$\begin{cases} \mathbf{z}^t = \sigma(\mathbf{W}_z\mathbf{x}^t + \mathbf{U}_z\mathbf{h}^{t-1}) \\ \mathbf{r}^t = \sigma(\mathbf{W}_r\mathbf{x}^t + \mathbf{U}_r\mathbf{h}^{t-1}) \\ \tilde{\mathbf{h}}^t = tanh(\mathbf{W}\mathbf{x}^t + \mathbf{U}(\mathbf{r}^t \odot \mathbf{h}^{t-1})) \\ \mathbf{h}^t = (1 - \mathbf{z}^t) \odot \mathbf{h}^{t-1} + \mathbf{z}^t \odot \tilde{\mathbf{h}}^t \end{cases}$$

The GRU can be treated as a black box. Given the current input $\mathbf{x}^t$ and previous hidden state $\mathbf{h}^{t-1}$, they merge two inputs and compute the current hidden state $\mathbf{h}^t$ in some way. This mechanism provides an effective way to preserve historical information for each node, and we can use it to encode the network evolution process.

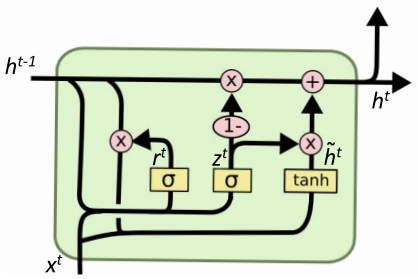


FIGURE 2. Framework of GRU [16].

B. ENCODER-DECODER OF DDNE

In this paper, we proposed a deep architecture to perform dynamic network embedding, whose framework is shown in Figure 3. In detail, to capture the historical dynamic pattern, we leverage a GRU to map the input sequence to a fixed-sized vector. This process can be treated as a GRU unfolds along with a series of time slices. We feed the input to the unit at each time slice and the unit calculates and updates a hidden state over time. Since GRU is known to learn problems with long range temporal dependencies and fast convergence, it makes the encoder efficient to capture the dynamic pattern by mapping the input to a highly non-linear latent space. However, the learning process of GRU takes each sample independently, while the correlations between samples have not been considered. To address this problem, we propose the interaction proximity to exploit the pairwise closeness between vertexes. Specifically, the proposed interaction proximity tries to map two frequently-connected vertexes into similar latent space. This can be achieved by optimizing the similarity of the hidden states between two vertexes. By using GRU units and computing interaction proximity, the encoder can preserve the historical information and capture the transitional patterns of dynamic networks. Additionally, we further design the decoder component to learn the embedding of future network, by leveraging the transitional information captured by encoder. For the decoder part, we extend a Deep Neural Network (DNN) to embed the output of encoder into a hidden layer. This hidden layer preserves all structure informations of new network and

condense each node's information into a k-dimension vector. The output of decoder is an inferable adjacency vector, which can be used to fit the new linkage state. We define a structure loss to describe the deviation between inference and new linkage. By jointly optimizing the interaction proximity and structure loss in the proposed encoder-decoder model, DDNE can preserve the highly-nonlinear dynamic network structure and generate the new networks embedding well. In the following section, we will introduce how to realize the supervised deep model in detail.

C. LOSS FUNCTIONS

The loss function for the supervised deep model has two components. We first introduce the structure loss, which exploits the nodes' transitional patterns to preserve and predict the network dynamic transition structures.

The structure loss refers to how precise the predicted structure is. Thus, to minimize this deviation, it is required to model neighborhood's transition of each vertex. Given a dynamic network $\mathcal{G}$, we first slice it evenly into several snapshots $\{\mathcal{G}_{t-N}, \ldots, \mathcal{G}_{t-1}, \mathcal{G}_t\}$. Then, we can obtain its adjacency matrix $\mathcal{X} = \{\mathbf{X}^{t-N}, \ldots, \mathbf{X}^{t-1}, \mathbf{X}^t\}$. For each adjacency matrix $\mathbf{X}^k, k \in [t-N, t]$, it contains n instances $\mathbf{X}^k(i,:), \ldots, \mathbf{X}^k(n,:)$. The value of each instance is $\mathbf{X}^k(i,:) = \{X_{ij}^k\}_{j=1}^n$, $X_{ij}^k > 0$ if and only if there exists connection between v_i, v_j during the time slice k. Therefore, there are various neighborhood structures for each vertex in a series of timestamps, and $\mathcal{X}$ provides the neighborhood structure and the temporal information of each node. With temporal adjacency matrices in $\mathcal{X}$, we adapt the encoder-decoder model to capture the transitional patterns and predict new network structure.

To further analyze the preserving and inference process, we briefly review the key idea of encoder-decoder model. As we emphasized in the last section, the deep model is composed of two parts, i.e. the encoder and the decoder. The encoder is a GRU that reads each symbol of an input sequence $\mathbf{x}$ sequentially. As it reads each symbol, the hidden state of the GRU changes as:

$$\mathbf{h}^t = f(\mathbf{h}^{t-1} + \mathbf{x}^t), \tag{1}$$

where f is a non-linear activation function. It is presented as an assemblage of GRU functions. After reading the end of the sequence, the output of encoder is a summary of all hidden states of the GRU. The decoder also consists of multiple non-linear functions, which maps the summary hidden states into representation space to infer network structure. Specifically, for one node i, given the temporal inputs $\mathbf{X}(i,:) = \{\mathbf{X}^{t-N}(i,:), \ldots, \mathbf{X}^{t-1}(i,:)\}$, the summary state for the encoder is shown as follows:

$$\mathbf{c}_i = [\mathbf{h}_i^{t-N}, \ldots, \mathbf{h}_i^{t-1}] \tag{2}$$

$$\mathbf{h}_i^k = [\overrightarrow{\mathbf{h}}_i^k, \overleftarrow{\mathbf{h}}_i^k], \quad k = \{t-N, \ldots, t-1\} \tag{3}$$

$$\overrightarrow{\mathbf{h}}_i^k = f(\overrightarrow{\mathbf{h}}_i^{k-1} + \overrightarrow{\mathbf{X}}^k(i,:)), \tag{4}$$

$$\overleftarrow{\mathbf{h}}_i^k = f(\overleftarrow{\mathbf{h}}_i^{k-1} + \overleftarrow{\mathbf{X}}^k(i,:)), \tag{5}$$

where $\overrightarrow{\mathbf{h}}_i^k$ and $\overleftarrow{\mathbf{h}}_i^k$ are based on Eq.(1), but they are fed with opposite time sequences as $\overrightarrow{\mathbf{X}}(i,:) = \{\mathbf{X}^{t-N}(i,:), \ldots, \mathbf{X}^{t-1}(i,:)\}$, $\overleftarrow{\mathbf{X}}(i,:) = \{\mathbf{X}^{t-1}(i,:), \ldots, \mathbf{X}^{t-N}(i,:)\}$. Two reversed hidden states are concatenated into $\mathbf{h}_i^k$, then all hidden states concatenated into the summary state $\mathbf{c}_i$. After obtaining $\mathbf{c}_i$, the hidden representations for each layer of decoder are presented as follows:

$$\mathbf{y}_i^{(1)} = \sigma(\mathbf{W}^{(1)}\mathbf{c}_i + \mathbf{b}^{(1)}) \tag{6}$$

$$\mathbf{y}_i^{(m)} = \sigma(\mathbf{W}^{(m)}\mathbf{y}_i^{(m-1)} + \mathbf{b}^{(m)}), \quad m = 2, \ldots, M \tag{7}$$

where M is the number of hidden layer. After the calculation process of decoder, we can obtain the output $\mathbf{y}_i^{(M)}$ as the new structure inference $\hat{\mathbf{X}}^t(i,:)$. The goal of decoder is minimizing the prediction error so that the output $\hat{\mathbf{X}}^t(i,:)$ can fit the linkage state $\mathbf{X}^t(i,:)$. We adopt cross entropy as the loss function, which is formulized as:

$$\begin{aligned}\ell_s &= -\sum_{i=1}^{n} \mathbf{X}^t(i,:) log \hat{\mathbf{X}}^t(i,:) \\ &= -\sum_{i=1}^{n}\sum_{j=1}^{n} X^t(i,j) \log \hat{X}^t(i,j)\end{aligned} \tag{8}$$

As a well-known assumption in dynamic network realms demonstrated, each vertex has a unique transitional pattern through time slices [3]. By mapping the relevant information to latent space, the encoder has the exponential capability to capture non-linear variance. Furthermore, by minimizing the structure loss, the decoder can use the transition information to perform embedding and infer structure of the new network.

However, such a supervised learning process cannot be directly applied to our problems because of sparsity of networks. In the real-world networks, we observed that most of the nodes have limited number of neighbors. Such a phenomenon can be found in the adjacency matrix, in which the number of non-zero elements is far less than that of zero elements. Thus, if we directly use $\mathbf{X}^t(i,:)$ as the learning target, the decoder is prone to predict the zero elements in the output vector. To address this problem, we impose more penalty to the prediction error of the non-zero elements than that of zero elements. Accordingly, the revised cross-entropy loss function is presented as:

$$\begin{aligned}\mathcal{L}_s &= -\sum_{i=1}^{n} \mathbf{Z}(i,:) \odot \mathbf{X}^t(i,:) log \hat{\mathbf{X}}^t(i,:) \\ &= -\sum_{i=1}^{n}\sum_{j=1}^{n} Z(i,j) X^t(i,j) \log \hat{X}^t(i,j)\end{aligned} \tag{9}$$

where $\mathbf{Z}(i,:) = \{Z(i,j)\}_{j=1}^n$. If $X(i,j) = 0$, $Z(i,j) = 1$, else $Z(i,j) = \alpha > 1$. Now by using the enhanced decoder model, the loss of non-zero elements can be highlighted and the linkage information will be preserved. Thus, the nodes' transitional patterns can be exploited to preserve and predict the network dynamic structure.

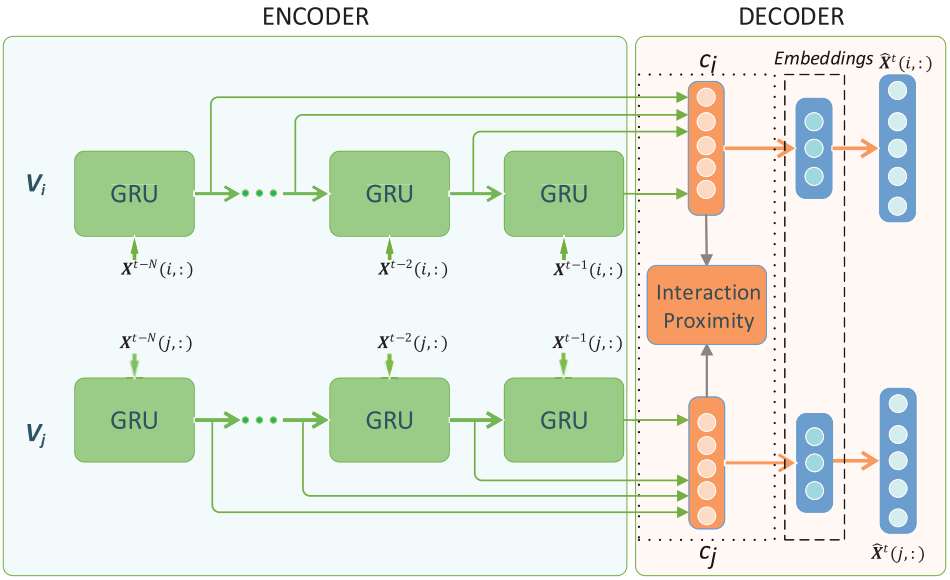


FIGURE 3. Framework of DDNE.

In a dynamic network, each node has a unique transitional pattern, however, a node's behavior is also influenced by its local neighbors [3]. How to quantify the influence and preserve network internal structure are important. We address this problem by embedding the interaction proximity into the deep learning structure. The interaction proximity can be regarded as the closeness of a pair of nodes, which is measured by using their links in the previous snapshots. Intuitively, two nodes that have frequent connections are more likely to have similar latent representations. To formulate this idea, we define the loss function as follows:

$$\mathcal{L}_c = \sum_{u,v=1}^{n} N_{ij} \|\mathbf{c}_i - \mathbf{c}_j\|_2 \tag{10}$$

where N_{ij} is the connection frequency of node i and j. If i, j have no edges in the historical network, $N_{ij} = 0$, else N_{ij} is amount of edges. We impose a penalty when similar nodes are mapped far away in the latent space. Thus, the frequently-connected nodes can have similar embeddings and tend to connect in the future network.

To capture the transition pattern and interaction proximity simultaneously, we propose a deep model to combine two loss functions and jointly minimize the following objective function:

$$\begin{aligned}\mathcal{L}_{all} &= \mathcal{L}_s + \beta\mathcal{L}_c + \gamma\mathcal{L}_{reg} \\ &= -\sum_{i=1}^{n} \mathbf{Z}(i,:) \odot \mathbf{X}^t(i,:) log\hat{\mathbf{X}}^t(i,:) \\ &\quad + \beta \sum_{i,j=1}^{n} N_{ij} \|\mathbf{c}_i - \mathbf{c}_j\|_2 + \gamma\mathcal{L}_{reg}\end{aligned} \tag{11}$$

where β and γ are hyper parameters to make a trade off all loss functions. $\mathcal{L}_{reg}$ is an $L2$-norm regularizer term to prevent overfitting problem, which is defined as follows:

$$\begin{aligned}\mathcal{L}_{reg} &= \|\mathbf{W}_z\|_F + \|\mathbf{W}_r\|_F + \|\mathbf{W}\|_F \\ &\quad + \|\mathbf{U}_z\|_F + \|\mathbf{U}_r\|_F + \|\mathbf{U}\|_F + \sum_{m=1}^{M} \|\mathbf{W}^{(m)}\|_F\end{aligned} \tag{12}$$

D. TRAINING AND INFERENCE ON DDNE

Since our proposed deep model has an encoder-decoder architecture, training and inference is no more difficult than in the sequence to sequence methods. We first make a forward propagation to calculate the loss, then back propagate the loss and update parameters to make model fit the inputs. In detail, we use the stochastic gradient descent (SGD) algorithm Adadelta [19] to automatically update and learn parameters. According to Eq.(1) to Eq.(7), the interaction proximity loss $\mathcal{L}_c$ can be summarized as:

$$\mathcal{L}_c = f_{en}(\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}, \mathbf{U}_z, \mathbf{U}_r, \mathbf{U}) \tag{13}$$

Then, the structure loss is:

$$\mathcal{L}_s = f_{all}(\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}, \mathbf{U}_z, \mathbf{U}_r, \mathbf{U}, \mathbf{W}^{(m)}, \mathbf{b}^{(m)}) \tag{14}$$

The regularizer function in Eq.(12) also can be presented as:

$$\mathcal{L}_{reg} = f_{reg}(\mathbf{W}_z, \mathbf{W}_r, \mathbf{U}, \mathbf{U}_z, \mathbf{U}_r, \mathbf{U}, \mathbf{W}^{(m)}) \tag{15}$$

Now, the updates for the weights $\mathbf{W}_z$, $\mathbf{W}_r$, $\mathbf{W}$, $\mathbf{U}_z$, $\mathbf{U}_r$, $\mathbf{U}$ have the same form as follows:

$$\mathbf{W}_z = \mathbf{W}_z - \lambda(\nabla f_{all}(\mathbf{W}_z) + \beta\nabla f_{en}(\mathbf{W}_z) + \gamma\nabla f_{reg}(\mathbf{W}_z)) \tag{16}$$

The parameters $\mathbf{W}^{(m)}$, $\mathbf{b}^{(m)}$ update as follows:

$$\begin{aligned}\mathbf{W}^{(m)} &= \mathbf{W}^{(m)} - \lambda(\nabla f_{all}(\mathbf{W}^{(m)}) + \gamma\nabla f_{reg}(\mathbf{W}^{(m)})) \\ \mathbf{b}^{(m)} &= \mathbf{b}^{(m)} - \lambda\nabla f_{all}(\mathbf{b}^{(m)}),\end{aligned} \tag{17}$$

TABLE 2. Statistics of networks.

Characteristics	arXiv	Enron	Radoslaw	Haggle
Nodes	4,122	11,670	151	91
Edges	220,349	91,971	2,869	1,142
Avg degree	106	15	38	25
Max degree	859	879	139	55
Clus coefficient	56.46%	25.14%	64.48%	80.09%

where λ is the learning rate, and $\nabla f(\mathbf{W})$ is the gradient of f at $\mathbf{W}$. We train the model following the update rules above. After that, model parameters are fixed and predicting future network status follows the similar reason with inferencing. Specifically, we can shift the window one step towards future to obtain a fixed observation which contains the previous $N-1$ snapshots and the current snapshot. We fix this known observation as history, put it into deep model and perform a forward inference to get the network embeddings at $t+1$. Now, these embeddings can be used to reconstruct network structure at $t+1$.

IV. EXPERIMENTS

In this section, we present both quantitative and qualitative experiment results on four real-word datasets. The experiment results demonstrate the effectiveness of the proposed DDNE model for dynamic network embedding.

A. DATASETS

In this paper, we use four datasets, including two email networks, one collaboration network and one human contact network, for two real-world applications, i.e. link prediction. They are available online in the Koblenz Network Collection (http://konect.uni-koblenz.de/). All networks have different sizes and attributes. Their statistic properties are shown in the Table 2.

The arXiv dataset [2], [20] is a collaboration graph of scientific paper authors from the arXiv's High Energy Physics - Phenomenology (hep-ph) section. This dataset has 10 years evolution history, ranging from 1991 to 2001, and the data increase steady in recent years. Hence, we chose five years (1995 - 1999) as five snapshots for our experiment. Each snapshot contains one-year network structure, and the last snapshot can be used as ground-truth of network inference.

Two email datasets (Enron and Radoslaw) contain email communication networks from two companies. For Enron email network [21], [22], it consists of 1,148,072 emails sent among employees of Enron between 1999 and 2003. For Radoslaw dataset [23], [24], it is the internal email communication network between employees of a mid-sized manufacturing company from 2010-01-01 to 2010-09-30. Nodes in both networks are individual employees and edges are individual emails. We sample five snapshots from Enron in every half year during 2001-01 to 2002-06 and denote them as $E1$ to $E5$. Radoslaw dataset is divided into nine slices ($R1$ to $R9$) ranging from January to September. We only extract five snapshots($R1$ to $R5$) for link prediction experiments.

The Haggle dataset [25], [26] is a real human contact network. This undirected network represents contacts between people measured by carried wireless devices. A node represents a person, and an edge between two persons shows that there was a contact between them. This dataset contains five-day records of 274 persons and we split it into five parts ($H1$ to $H5$) based on each day.

To summarize, we adopt five snapshots of temporal datasets for link prediction task. The first snapshot is used to construct the basic network, which contains all the nodes through all time slices. Snapshots 2, 3 and 4 are used for training models. After model training, we move one step forward, snapshots 3, 4 now can be adopted to inference the embedding and structure of snapshot 5. The ground-truth network structure in the last snapshot is used to validate inference result.

B. EVALUATION SCHEME

For link prediction task, we are trying to leverage historical linkage information to infer the embedding and structure of current network. It is essentially a binary classification problem. Given n nodes, we try to predict which pair of nodes will generate an edge. However, only a very small fraction of links actually exists, which will lead to a data imbalance problem.

Our experiments show that the existing links only constitute less than 1% of all possible links. This means that if we set all prediction result to zero, we can still achieve a high accuracy evaluation. Thus, in order to evaluate the performance properly, we use the following measurement:

- Area Under the ROC Curve (AUC): it is frequently employed on classification problem because it relates to the sensitivity (true positive rate) and the specificity (true negative rate) of a classifier. This metric is strictly bounded between 0 and 1. The larger the AUC is, the better the model performs.
- Mean Average Precision (MAP): mean average precision is an extension of average precision (AP) where we take average of all AP's to get the MAP. It estimates precision for every node and computes the average over all nodes. Compared with AUC, it considers whether all of the relevant items tend to get ranked highly. It is calculated as follows:

$$MAP = \frac{\sum_1^n AP(i)}{|V|} \tag{18}$$

$$AP(i) = \frac{\sum_j precision@j(i) \cdot \Delta_i(j)}{|\{\Delta_i(j) = 1\}|} \tag{19}$$

where $precision@j(i) = \frac{|E_{pred_i}(1:j) \cap E_{obs_i}|}{j}$, and E_{pred_i} and E_{obs_i} are the predicted and observed edges for node i respectively. $\Delta_i(j) = 1$ indicates that v_i and v_j have a link. V is the query set for information retrieval. In our experiments, it represents a node set.

C. BASELINE METHODS

We use various methods as the baselines, which include unsupervised learning methods such as Common Neighbors, Katz [27], generative models tRBM and ctRBM, and network embedding algorithms such as DeepWalk, node2vec and SDNE. We make a full comparison with these baselines to show the capability of our embedding method in the task of link prediction. The brief introduction of baselines is listed as follows:

- Common Neighbors (CN): The CN metric is one of the most widespread measurements used in link prediction problem. A large number of the common neighbors make it easier to have a link between two nodes.
- Katz [27]: It is based on the ensemble of all paths, and it counts all paths with different weights between two nodes.
- temporal Restricted Boltzmann Machine (tRBM) [12] : A RBM based model which takes adjacency matrixes as inputs. It can be used to describe network's dynamics and predict network structure.
- conditional temporal Restricted Boltzmann Machine (ctRBM) [3]: A tRBM based model embeds the information of neighbor nodes.
- DeepWalk [11]: It adopts random walk and skip-gram model to generate network representations. The represented embeddings can be used to predict linkage state of network.
- node2vec [10]: Similar to DeepWalk, node2vec preserves higherorder proximity between nodes by maximizing the probability of occurrence of subsequent nodes in fixed length random walks [28].
- SDNE [9]: It is an autoencoder based deep model, which defines loss functions of the first-order or second-order proximity to preserve network internal and network dynamic transition structures.

D. PARAMETER SETTINGS

We proposed a deep model in this paper, which can represent network into a low dimension space. It has different embedding size with different datasets. For the small datasets such as Radoslaw (151 nodes) and Haggle (91 nodes), we set the output has the same size as the input. For the large datasets such as ArXiv (4122 nodes) and Enron(11670 nodes), the dimension of output is set to 1024. If we use higher dimensionality, the performance almost remains unchanged or even becomes worse. The hyper-parameters of α, β and γ are tuned by using grid search on the validation set. For different datasets, the parameters for baselines are different, and all are tuned to be optimal. Other default settings include: the learning rate of deep model is set as 0.0001; the history window size is set to 2 so that we can infer the embedding by two historical snapshots.

E. EXPERIMENTAL RESULTS

After setting the parameters, we conduct the link prediction task in four real-world networks. We first leverage historical snapshots and current linkage status to train DDNE model. After training, as all parameters are adapted to the dataset, we shift the window one step towards future to obtain a fixed observation which contains the previous $N-1$ snapshots and the current snapshot. Now, the DDNE is performed as a generative model. We fix this known observation as history, and put it into deep model, such that we can obtain the representations for each vertex and then use the obtained representations to predict the network status at $t+1$. For the generative models such as tRBM and ctRBM, we adopt the same strategy for the experiments. For embedding methods such as DeepWalk, node2vec, SDNE, a combined historical network that merges all previous snapshots into one graph is directly employed. As Wang *et al.* [9] and Grover and Leskovec [10] demonstrated, they can learn the representations of each vertex and then make predictions for the unobserved links.

TABLE 3. AUC on different datasets of the link prediction.

Approaches	arXiv	Enron	Radoslaw	Haggle
CN	0.8115	0.7551	0.8636	0.9621
Katz	0.8117	0.7581	0.8676	0.9658
tRBM	0.8476	0.8129	0.8478	0.9611
ctRBM	0.7997	0.7262	0.8645	0.9618
DeepWalk	0.6071	0.5230	0.5308	0.7269
node2vec	0.8341	0.9349	0.7813	0.7864
SDNE	0.8146	0.9437	0.8709	0.8452
DDNE-w/o IP	0.8809	0.9532	0.8741	0.9577
DDNE	**0.9002**	**0.9602**	**0.9465**	**0.9736**

Table 3 compares AUC performances over the four datasets described in Section IV.A, in which DDNE-without IP is the proposed model without Interaction Proximity. The result shows that DDNE achieves the best among all baselines regardless whether the network is very sparse (Enron) or dense (Radoslaw, Haggle). It is worth noting that conventional embedding methods do not perform better than other baselines, since they only preserve the structure information but ignore the transitional information of dynamic networks. In the proposed method, both the structure information and transition of networks are well considered so that it is capable to infer the new networks' status. Furthermore, we find that the DDNE has better performances than the DDNE-without IP, which indicates the interaction information effectively facilitates the embedding and inference of new network. To further specify the AUC results of DDNE model, we illustrate the Receiver Operating Characteristic (ROC) on four datasets. The illustration is shown in Figure 4. For each dataset, there are different number of thresholds since they have different amount of samples. The area under the ROC curve equals to the AUC result, which is presented in Table 3. We adopt sklearn [29] to complete the ROC calculation. It worth noting that sklearn sometimes decides to drop some non-useful thresholds, resulting in thresholds to be less than distinct values.

Table 4 reports MAP performances on link prediction. Similar results can be achieved as in the AUC evaluation.

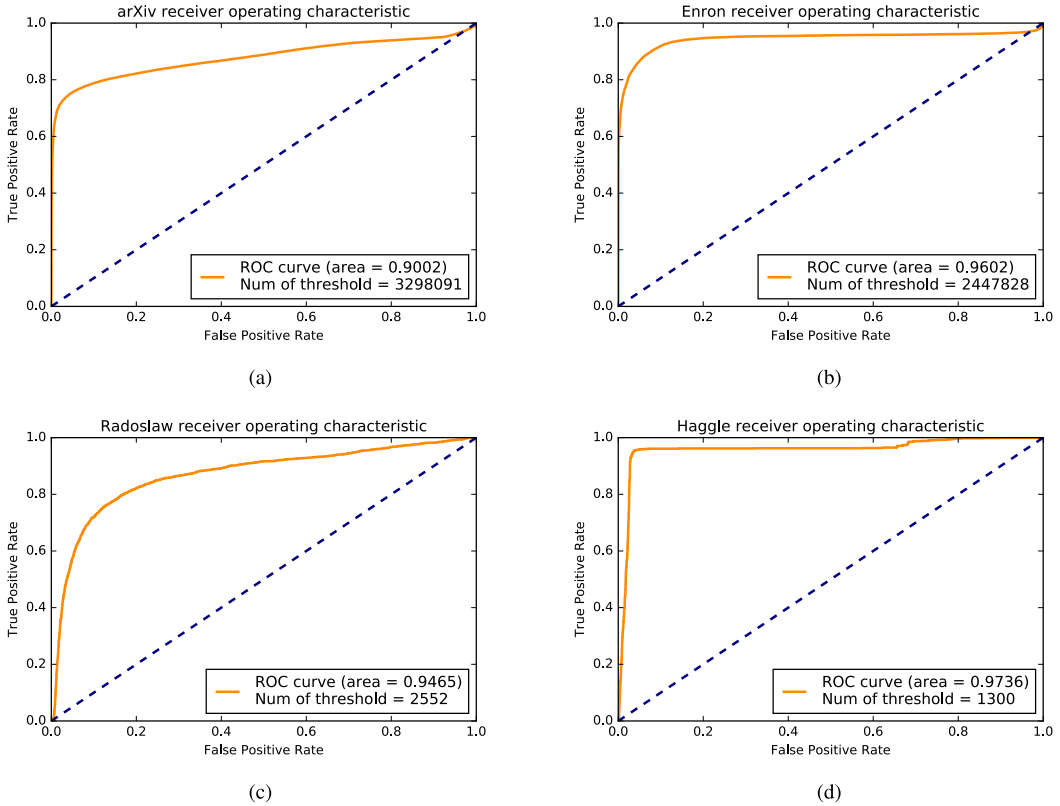


FIGURE 4. Experiments on ROC curve. (a) ROC curve on arXiv. (b) ROC curve on Enron. (c) ROC curve on Radoslaw. (d) ROC curve on Haggle.

TABLE 4. MAP on different datasets of the link prediction.

Approaches	arXiv	Enron	Radoslaw	Haggle
CN	0.1802	0.0169	0.2361	0.3798
Katz	0.1926	0.0182	0.392	0.4429
tRBM	0.2013	0.0278	0.3292	0.4035
ctRBM	0.2324	0.0228	0.3257	0.3995
DeepWalk	0.0127	0.0003	0.1033	0.3919
node2vec	0.1219	0.044	0.3259	0.521
SDNE	0.0287	0.0403	0.2816	0.4403
DDNE-w/o IP	0.3167	**0.2065**	0.4463	0.6054
DDNE	**0.3464**	0.1954	**0.5377**	**0.733**

The proposed method achieves substantial gains than benchmarks. An interest finding is that DDNE-without IP performs slightly better than DDNE on Enron dataset. This is because in the sparse network, the user interactions are rare through all snapshots, DDNE may not capture these interactions in training process.

To further explore the experimental results, we conduct a statistical analysis to examine whether the DDNE's performance is significantly different from baselines. Generally, there are several tests to investigate statistical significance, such as student's t-test, ANOVA tests etc. We decide to use two-sample t-test since it can determine whether two population means are different. Specifically, we first conduct 30-time experiments for each method and each dataset so that we obtain 30 samples for a method-dataset pair.

TABLE 5. P value analysis about AUC.

p-value	arXiv DDNE	Enron DDNE	Radoslaw DDNE	Haggle DDNE
CN	7.24E-109	7.22E-53	2.60E-28	0.0060
Katz	8.26E-109	1.67E-52	3.21E-27	**0.0580**
tRBM	1.16E-18	9.50E-35	7.85E-24	0.0206
ctRBM	4.44E-34	1.07E-47	2.29E-20	0.0385
DeepWalk	2.16E-64	1.91E-61	4.00E-59	2.60E-45
node2vec	4.96E-24	1.86E-05	7.02E-37	1.87E-39
SDNE	2.37E-31	0.0014	1.48E-19	2.18E-29

Then, we adopt t-test to compare our proposed model DDNE with each baseline. The output of t-test is a p-value, which indicates the strength of observations against null hypothesis. In our experiments, a small p-value (typically ≤ 0.05) indicates that the means of two method-dataset results are significantly different. Table 5 and Table 6 present the statistical results of two metrics. Some elements in the table have the type like $xE-y$, which means $x * 10^{-y}$. From the tables, we can see that except for the AUC result with DDNE and Katz (p-value=0.058), the proposed model performs significantly different than all baselines in terms of four datasets. These p-values also give strong backing to the AUC and MAP results in Table 3 and Table 4.

F. PARAMETER SENSITIVITY ANALYSIS

We further investigate the parameter sensitivity in this section. Specifically, we evaluate how different length of

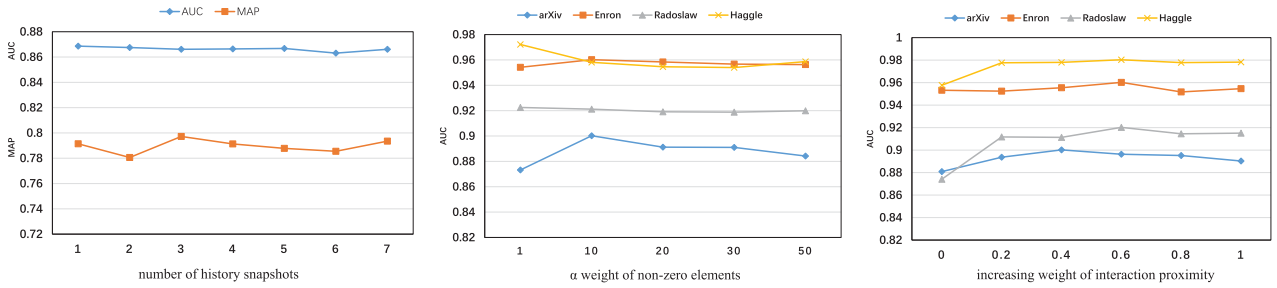


FIGURE 5. Experiments on parameters sensitivity. (a) The performances on Radoslaw with increasing historical snapshots. (b) The performances on four datasets with increasing α. (c) The performances on four datasets with increasing β.

TABLE 6. P value analysis about MAP.

p-value	arXiv DDNE	Enron DDNE	Radoslaw DDNE	Haggle DDNE
CN	1.65E-45	4.40E-56	1.23E-66	2.31E-62
Katz	1.28E-43	6.68E-56	1.52E-48	1.87E-57
tRBM	5.76E-36	1.75E-41	2.50E-49	5.41E-55
ctRBM	2.66E-28	1.11E-38	8.48E-44	2.16E-53
DeepWalk	5.79E-55	1.02E-45	1.35E-62	4.88E-54
node2vec	1.77E-45	2.72E-42	1.91E-47	1.23E-43
SDNE	3.08E-51	1.33E-38	1.28E-53	6.02E-52

the historical time and different values of hyper-parameter α and β can affect the link prediction results.

1) INFLUENCE OF HISTORY LENGTH

In our work, three datasets are split into five snapshots, while only Radoslaw dataset has nine time slices. Thus, we conduct this experiment on Radoslaw to examine how the different choices of parameters affect the performance of DDNE. We vary the number of historical snapshots from 1 to 7, to demonstrate the effect of varying this parameter. Except for the parameter being tested, all other parameters assume default values. The result shows that DDNE is not very sensitive to the number of historical snapshots. As we can see in Figure 5(a) there is no significant accuracy improvement after the number of snapshots increases. The reason is that a small dense network may have similar linkage states in each snapshot, and the increment of historical links does not enrich the information for network inference.

2) INFLUENCE OF HYPER PARAMETER α

In our model, we set α as a hyper parameter to control the inference weight of the non-zero elements in training graph. The larger the α, the model will more prone to predict the non-zero elements. The result is shown in Figure 5(b). For the sparse networks such as ArXiv and Enron, we can see that the performance raises and then keep stable when the weight of the non-zero elements increases. This is intuitive as the model pay more attention on linked nodes so that it can preserve linkage information among tremendous non-link node pairs. However, for the dense networks (Radoslaw and Haggle), the performance has no improvement and even being worse when the weight α increases. This is because the non-zero and zero elements in training network is balanced, too large a weight on non-zero elements may overwhelm the zero elements and also introduce noises to deteriorate the performance. Therefore, it is always important to determine the weight of non-zero elements for each type of networks.

3) INFLUENCE OF HYPER PARAMETER β

The parameter of β balances the weight of the structure loss and interaction proximity. When $\beta = 0$, the performance is totally determined by structure loss. The larger the β, the more the model concentrates on interaction proximity. When $\beta = 1$, both loss functions have same contributions to the performance. As we can see from Figure 5(c), the evaluation metric AUC raises when the parameter β increases. However, when the weight of interaction proximity continuously increases, the performance starts to drop slowly. The reason is that too much interaction proximity may overwhelm the structure loss, and then it deteriorates the performance. It also demonstrates that both structure loss and interaction proximity are essential for network embedding methods to characterize and infer the network structure.

V. CONCLUSIONS

In this paper, we propose a Deep Dynamic Network Embedding, namely DDNE, for link prediction task. Specifically, to model the evolving pattern of each node, we design a new deep architecture, which can leverage historical linkage to make an embedding for new links. To further address the neighbor's influence problem, we exploit the interaction proximity in the hidden state to measure the similarity of nodes. By jointly optimizing them and put more weight on non-zero elements of output, the learned embeddings are new-linkage-preserved and are robust to sparse networks. Empirically, we compare the proposed model with traditional embedding methods and state-of-the-art link prediction methods in a variety of datasets. Experiment results demonstrate that we achieve significant gains than other baselines.

Our future work will focus on how to learn representations for heterogeneous networks which have different type of nodes and edges. Furthermore, we will try to reduce the computation complexity of the proposed deep model.

REFERENCES

[1] M. Spiliopoulou, "Evolution in social networks: A survey," in *Social Network Data Analytics*. Boston, MA, USA: Springer, 2011, pp. 149–175.

[2] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Discovery Data*, vol. 1, no. 1, pp. 1–40, 2007.

[3] X. Li, N. Du, H. Li, K. Li, J. Gao, and A. Zhang, "A deep learning approach to link prediction in dynamic networks," in *Proc. SIAM Int. Conf. Data Mining*, 2014, pp. 289–297.

[4] L. Tang, H. Liu, J. Zhang, and Z. Nazeri, "Community evolution in dynamic multi-mode networks," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 677–685.

[5] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.

[6] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, 2003.

[7] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web Int. World Wide Web Conf. Steering Committee*, 2015, pp. 1067–1077.

[8] D. Luo, C. H. Q. Ding, F. Nie, and H. Huang, "Cauchy graph embedding," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 553–560.

[9] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1225–1234.

[10] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.

[11] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 701–710.

[12] G. W. Taylor, G. E. Hinton, and S. T. Roweis, "Modeling human motion using binary latent variables," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 1345–1352.

[13] D. Bahdanau, K. Cho, and Y. Bengio. (2014). "Neural machine translation by jointly learning to align and translate." [Online]. Available: https://arxiv.org/abs/1409.0473

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[15] K. Cho *et al.* (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation." [Online]. Available: https://arxiv.org/abs/1406.1078

[16] (2015). *Understanding LSTM Networks*. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs

[17] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. (2014). "On the properties of neural machine translation: Encoder-decoder approaches." [Online]. Available:https://arxiv.org/abs/1409.1259

[18] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling." [Online]. Available: https://arxiv.org/abs/1412.3555

[19] M. D. Zeiler. (2012). "ADADELTA: An adaptive learning rate method." [Online]. Available: https://arxiv.org/abs/1212.5701

[20] (Oct. 2016). *Arxiv Hep-Ph Network Dataset–KONECT*. [Online]. Available: http://konect.uni-koblenz.de/networks/ca-cit-HepPh

[21] (Oct. 2016). *Enron Network Dataset–KONECT*. [Online]. Available: http://konect.uni-koblenz.de/networks/enron

[22] B. Klimt and Y. Yang, "The Enron corpus: A new dataset for email classification research," in *Proc. Eur. Conf. Mach. Learn.*, 2004, pp. 217–226.

[23] (Oct. 2016). *Manufacturing Emails Network Dataset–KONECT*. [Online]. Available: http://konect.uni-koblenz.de/networks/radoslaw_email

[24] R. Michalski, S. Palus, and P. Kazienko, "Matching organizational structure and social network extracted from email communication," in *Business Information Systems* (Lecture Notes in Business Information Processing), vol. 87. Berlin, Germany: Springer, 2011, pp. 197–206.

[25] (Oct. 2016). *Haggle Network Dataset–KONECT*. [Online]. Available: http://konect.uni-koblenz.de/networks/contact

[26] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of human mobility on opportunistic forwarding algorithms," *IEEE Trans. Mobile Comput.*, vol. 6, no. 6, pp. 606–620, Jun. 2007.

[27] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[28] P. Goyal and E. Ferrara. (2017). "Graph embedding techniques, applications, and performance: A survey." [Online]. Available: https://arxiv.org/abs/1705.02801

[29] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

TAISONG LI was born in Chenzhou, China, in 1990. He received the B.E. degree in communication engineering from Jilin University, Jilin, China, in 2013. He is currently pursuing the Ph.D. degree with the Institute of Acoustics, University of Chinese Academy of Sciences, Beijing, China.

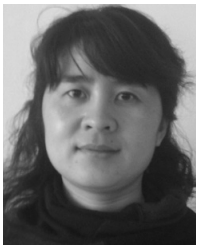
JIAWEI ZHANG founded IFM Lab in 2017, which is a research oriented academic laboratory, providing the latest information on fusion learning and data mining research works and application tools to both academia and industry, where he has been the Director since 2018. He is currently an Assistant Professor with the Department of Computer Science, Florida State University.

He received the B.S. degree from the Department of Computer Science, Nanjing University, in 2012, and the Ph.D. degree in computer science from The University of Illinois at Chicago in 2017. He joined Yahoo! Research in 2016, the IBM T. J. Watson Research Center in 2015, and Microsoft Research in 2014, as Summer Research Intern.

PHILIP S. YU (S'76–M'78–SM'87–F'93) was with the IBM Thomas J. Watson Research Center, where he was the Manager of the Software Tools and Techniques Department. He is currently a Distinguished Professor with the Department of Computer Science, UIC, where he is also the Wexler Chair of information and technology. He has authored over 970 papers in refereed journals and conferences with over 74,500 citations and an H-index of 127. He holds or applied for over 300 U.S. patents. His main research interests include big data, data mining (especially on graph/network mining), social network, privacy preserving data publishing, data stream, database systems, and Internet applications and technologies.

Dr. Yu was a recipient of the ACM SIGKDD 2016 Innovation Award for his influential research and scientific contributions on mining, fusion, and anonymization of big data, the IEEE Computer Society's 2013 Technical Achievement Award for "pioneering and fundamentally innovative contributions to scalable indexing, querying, searching, mining, and anonymization of big data," and the IEEE Region 1 Award for "promoting and perpetuating numerous new electrical engineering concepts" in 1999. He received the Research Contributions Award from the IEEE International Conference on Data Mining in 2003 for his pioneering contributions to the field of data mining and. He received several UIC honors, including the Research of the Year at 2013 and the UI Faculty Scholar at 2014 and many IBM honors including two IBM outstanding innovation awards, an Outstanding Technical Achievement Award, two research division awards, and the 94th Plateau of Invention Achievement Award. He was an IBM Master Inventor. He is a fellow of the ACM.

He received the B.S. degree in electrical engineering from National Taiwan University, the M.S. and Ph.D. degrees in electrical engineering from Stanford University, and the M.B.A. degree from New York University.



YAN ZHANG was born in Beijing, China, in 1973. She received the B.E. and M.S. degrees in automatic control from the Beijing Institute of Technology, China, in 1996 and 1999, respectively, and the Ph.D. degree in pattern recognition and intelligent system from the Institute of Automation, Chinese Academy of Sciences, Beijing, in 2003. From 2003 to 2006, she was a Researcher with the Spoken Language Communication Research Laboratories, Advanced Telecommunications Research Institute International, Japan. She was an Associate Professor with the Institute of Acoustics (IOA), Chinese Academy of Sciences (CAS). Since 2006, she has been a Researcher with the Research and Development Center, Toshiba Co., Ltd, China. Since 2011, she has been with IOA, CAS.

Her main areas of research interest are natural language processing, social media processing, and big data analysis.

YONGHONG YAN received the bachelor's degree from the Electronic Engineering Department, Qinghua University, in 1990, and the Ph.D. degree in computer science and engineering from the Oregon Graduate Institute of Science and Engineering (OGI) in 1995. He was the Director of the division engineers of the Intel Microprocessors Laboratory and focused on computer interface general framework, the Director and the Principal Engineer of the Intel China Research Center, the Intel Global Computer Interface Academic Chairman of the Committee, and an Associate Professor of Oregon Health and Science University. In 2002, he was elected in "CAS scientists." Since 1995, he has been teaching master's and doctoral students with the Oregon Research Institute and has been an Associate Director of the Center of Spoken Language Understanding, OGI. He is currently a Researcher with the Chinese Academy of Sciences Institute of Acoustic, a Tutor of doctoral students, and the Director and an Assistant Director of the Laboratory ThinkIT. He served as the Chinese National Natural Science Fund Committee and the U.S. Expert Panel and the National "242" Panel of Experts, including a number of members post.