E-LSTM-D: A Deep Learning Framework for Dynamic Network Link Prediction

Jinyin Chen, Jian Zhang, Xuanheng Xu, Chenbo Fu, Dan Zhang, Qingpeng Zhang, *Member, IEEE*, and Qi Xuan, *Member, IEEE*

***Abstract*—Predicting the potential relations between nodes in networks, known as link prediction, has long been a challenge in network science. However, most studies just focused on link prediction of static network, while real-world networks always evolve over time with the occurrence and vanishing of nodes and links. Dynamic network link prediction (DNLP) thus has been attracting more and more attention since it can better capture the evolution nature of networks, but still most algorithms fail to achieve satisfied prediction accuracy. Motivated by the excellent performance of long short-term memory (LSTM) in processing time series, in this article, we propose a novel encoder-LSTM-decoder (E-LSTM-D) deep learning model to predict dynamic links end to end. It could handle long-term prediction problems, and suits the networks of different scales with fine-tuned structure. To the best of our knowledge, it is the first time that LSTM, together with an encoder–decoder architecture, is applied to link prediction in dynamic networks. This new model is able to automatically learn structural and temporal features in a unified framework, which can predict the links that never appear in the network before. The extensive experiments show that our E-LSTM-D model significantly outperforms newly proposed DNLP methods and obtain the state-of-the-art results.**

***Index Terms*—Dynamic network, encoder–decoder, link prediction, long short-term memory (LSTM), network embedding.**

Manuscript received January 18, 2019; revised May 5, 2019; accepted July 17, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61502423, Grant 61973273, Grant 71672163, and Grant 61572439, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LR19F030001 and Grant LY19F020025, in part by the Zhejiang Science and Technology Plan Project under Grant LGF18F030009, and in part by the Key Technologies, System and Application of Cyberspace Big Search, Major Project of Zhejiang Laboratory under Grant 2019DH0ZX01. This article was recommended by Associate Editor W. Hsu. *(Corresponding author: Qi Xuan.)*

J. Chen is with the Institute of Cyberspace Security, College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China, and also with the Big Search in Cyberspace Research Center, Zhejiang Laboratory, Hangzhou 311121, China (e-mail: chenjinyin@zjut.edu.cn).

J. Zhang, X. Xu, C. Fu, and D. Zhang are with the Institute of Cyberspace Security, College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China (e-mail: zj_1994@outlook.com; 2111603112@zjut.edu.cn; cbfu@zjut.edu.cn; danzhang@zjut.edu.cn).

Q. Zhang is with the School of Data Science, City University of Hong Kong, Hong Kong (e-mail: qingpeng.zhang@cityu.edu.hk).

Q. Xuan is with the Big Search in Cyberspace Research Center, Zhejiang Laboratory, Hangzhou 311121, China, and also with the Institute of Cyberspace Security, College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China (e-mail: xuanqi@zjut.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2019.2932913

I. Introduction

NETWORKS are often used to describe complex systems in various areas, such as social science [1], [2], biology [3], electric system [4], economics [5], etc. And the vast majority of the real-world systems evolve with time, which can be modeled as dynamic networks [6], [7], where the nodes may come and go and the links may vanish and recover as time goes by. Links, representing the interactions between different entities, are of particular significance in the analysis of dynamic networks.

Link prediction of a dynamic network [8], [9] tries to predict the future structure of the network based on the historical data, which helps us better understand network evolution and further the relationships between topologies and functions. For instance, in online social networks [10]–[12], we can predict which links are going to be established in the near future. It means that we can infer with what kind of people, or even which particular one, the target user probably makes friends base on their historical behaviors. It can also be applied to the studies on disease contagions [13], protein–protein interactions [14], and many other fields where the evolution matters.

Similarity indices, like common neighbor (CN) [15] and resource allocation (RA) index [16], have widely used in link prediction of static networks [17]. Though simple and sometimes effective, they can not deal with the high nonlinearity. Fortunately, recently developed graph embedding methods are going to solve this problem. From random walk-based approaches like DeepWalk [18] and node2vec [19] to deep learning-based models like structural deep network embedding (SDNE) [20] and graph convolution network (GCN) [21], graph embedding techniques project nodes, links, or even graphs into lower dimensional space, ensuring that the entities sharing similar characteristics stay close, and thus could efficiently improve the performance of link prediction algorithms. However, the methods above still lack of applicability when it comes to dynamic networks. The temporal information is always ignored by the models designed for static networks. A few of works contribute to making above methods adapt to dynamic networks, such as redefining CNs [22] and designing temporal walk for dynamic networks [23], [24]. And with the development of deep learning [25], [26], a bunch of end-to-end frameworks [27], [28] have been proposed for dynamic network link prediction (DNLP). Not only limited to homogeneous networks, several works [29], [30] have explored the link prediction problem on heterogeneous dynamic networks.

In this article, we address the problem of predicting the global structure of networks in the near future, focusing on the links that are going to appear or disappear. We propose a novel end-to-end encoder-LSTM-decoder (E-LSTM-D)[1] deep learning model for link prediction in dynamic networks, which takes the advantages of encoder–decoder architecture and a stacked long short-term memory (LSTM) [31]. The model thus can effectively handle the problems of high dimension, nonlinearity, and sparsity. Due to the encoder–decoder architecture, the model can automatically learn representations of networks, as well as reconstruct a graph on the grounds of the extracted information. Relatively low-dimensional representations for the sequences of graphs can be well learned from the stacked LSTM module placed right behind the encoder. Considering that network sparsity may seriously affect the performance of the model, we amplify the effect of existing links at the training process, enforcing the model to account for the existing links more than missing/nonexistent ones. We conduct comprehensive experiments on five real-world datasets. The results show that our model significantly outperforms the current state-of-the-art methods. In particular, we make the following main contributions.

1) We propose a general end-to-end deep learning framework, namely E-LSTM-D, for link prediction in dynamic networks, where the encoder–decoder architecture automatically learns representations of networks and the stacked LSTM module enhances the ability of learning temporal features.
2) Our newly proposed E-LSTM-D model is competent to make long term prediction tasks with only slight drop of performances; It suits the networks of different scales by fine tuning the model structure, i.e., changing the number of units in different layers; Besides, it can predict the links that are going to appear or disappear, while most existing methods only focus on the former.
3) We define a new metric, *error rate*, to measure the performance of DNLP, which is a good addition to the area under the ROC curve (AUC), so that the evaluation is more comprehensive.
4) We conduct extensive experiments, comparing our E-LSTM-D model with five baseline methods on various metrics. It is shown that our model outperforms the others and obtain the state-of-the-art results.

The rest of this article is organized as follows. In Section II, we give a brief review of recent works on DNLP. In Section III, we provide a rigorous definition of DNLP and a detailed description of our E-LSTM-D model. After that, comprehensive experiments are presented in Section IV, with the results carefully discussed. Finally, we conclude this article and outline some future works in Section V.

II. Related Work

A. Random Walk-Based Methods

Recent studies show that random walk can also be performed on dynamic networks [32]. Ahmed *et al.* [33] assigned damping weights to each snapshots, ensuring that more recent snapshots are more important, and then combined them into a weighted graph to do local random walk. As an extension of [33], Ahmed and Chen [23] proposed time series random walk (TS-RW) to integrate temporal and global information. And in [34], random walk was used as a sampling method before passing the network into the deep neural network. Nguyen *et al.* [24] studied continuous-time dynamic networks rather than discrete snapshots and proposed temporal walk to learn time-dependent embedding vectors without loss of information.

B. Factorization-Based Methods

Incremental singular value decomposition (IncSVD) [35] utilizes a perturbation matrix to represent dynamics and adds modification on the SVD while theoretically instructed maximum-error-bounded restart of SVD (TIMERS) [36] focuses on the adjacent matrix. Ma *et al.* [37] proposed a novel graph regularized non-negative matrix factorization algorithm (GrNMF). It factorizes the current graph with the previous graphs used as regularization.

C. Deep Learning-Based Methods

Considering the evolution of a network as a special case of Markov random field with two-layer variables, conditional temporal restricted Boltzmann machine (ctRBM) [38] takes not only neighboring connections but also temporal connections into consideration, and thus has the ability to predict future linkage status. In fact, the neighborhood information could be better used. Li *et al.* [39] incorporated temporal restricted Boltzmann machine (RBM) and gradient boosting decision tree (GBDT) to model each link's evolution pattern. Apart from the methods based on RBM, another group of approaches are built upon recurrent neural networks (RNNs). Li *et al.* [27] designed a two-stream encoder–decoder architecture and used gated recurrent unit (GRU) [40] as encoder to learn both spatial and temporal information. Similarly, DynGEM [41] utilizes deep auto-encoders to incrementally generate embedding vectors for the graph of next snapshot based on the current one.

D. Other Methods

With the improvements of traditional similarity indices, Yao *et al.* [22] assigned time-varied weights to previous graphs and then executed link prediction task using the refined CN which considers the neighbors within two hops. Zhang *et al.* [42] proposed an improved RA-based DNLP algorithm, which updates the similarity between pairwise nodes when the network structure changes. Besides, Zhou *et al.* [43] modeled the network evolution as a triadic closure process, but limited to undirected networks.

These methods either take a pair of nodes as input while ignore the global structure of the whole snapshot, or just make use of a few historical snapshots while ignore the rich information contained in the snapshots early before. In this article, we propose our E-LSTM-D model which takes a relatively long sequence of snapshots as input and could automatically learn the global structure of networks.

[1]The source code is open sourced at https://github.com/jianz94/e-lstm-d.

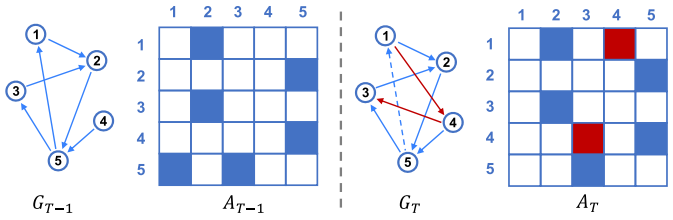


Fig. 1. Illustration of network evolution. The structure of the network changes overtime. At time T, $E_{(1,4)}$ and $E_{(4,3)}$ emerge while $E_{(5,1)}$ vanishes, which is reflected in the change of A, with those elements equal to 1 represented by filled squares.

III. Methodology

In this section, we will introduce our E-LSTM-D model used to predict the evolution of dynamic networks.

A. Problem Definition

A dynamic network is modeled as a sequence of snapshot graphs taken at a fixed interval.

Definition 1 (Dynamic Networks): Given a sequence of graphs, $\{G_1,\ldots, G_T\}$, where $G_k = (V, E_k)$ denotes the kth snapshot of a dynamic network. Let V be the set of all vertices and $E_k \subseteq V \times V$ the temporal links within the fixed timespan $[t_{k-1}, t_k]$. The adjacency matrix of G_k is denoted by A_k with the element $a_{k;i,j} = 1$ if there is a directed link from v_i to v_j and $a_{k;i,j} = 0$ otherwise.

In a static network, link prediction aims to find edges that actually exist according to the distribution of observed edges. Similarly, link prediction in a dynamic network makes full use of the information extracted from previous graphs to reveal the underlying network evolving patterns, so as to predict the future status of the network. Since the adjacency matrix can precisely describe the structure of a network, it is ideal to use it as the input and output of the prediction model. We could infer G_t just based on G_{t-1}, due to the strong relationship between the successive snapshots of the dynamic network. However, the information contained in G_t may be too little to do precise inference. In fact, not only the structure itself but also the structure change overtime matters in the network evolution. Thus, we prefer to use a sequence of length N, i.e., $\{G_{t-N},\ldots, G_{t-1}\}$, to predict G_t.

Definition 2 (Dynamic Network Link Prediction): Given a sequence of graphs with length N, $S = \{G_{t-N},\ldots, G_{t-1}\}$, DNLP aims to learn a function that maps the input sequence S to G_t.

The structure of a dynamic network evolves with time. As shown in Fig. 1, some links may emerge while some others may vanish, which can be reflected by the changes of the adjacency matrix overtime. The goal is to find the links of the network that are most likely to appear or disappear at the next timespan. Mathematically, it can also be interpreted as an optimization problem of finding a matrix, whose element is either 0 or 1, that can best fit the ground truth.

B. E-LSTM-D Framework

Here, we propose a novel deep learning model, namely E-LSTM-D, combining the architecture of encoder–decoder and stacked LSTM, with the overall framework shown in Fig. 2. Specifically, the encoder is placed at the entrance of the model to learn the highly nonlinear network structures and the decoder converts the extracted features back to the original space. Such encoder–decoder architecture is capable of dealing with spatial nonlinearity and sparsity, while the stacked LSTM between the encoder and decoder can learn temporal dependencies. The well designed end-to-end model thus can learn both structural and temporal features and do link prediction in a unified way.

TABLE I
Terms and Notations Used in the Framework

Symbol	Definition
K	number of encoder/decoder layers
L	number of LSTM cells
$\hat{A}_t$	output of the decoder
H	output of the stacked LSTM
$Y_e^{(k)}$, $Y_d^{(k)}$	output of k^{th} encoder/decoder layer
$W_e^{(k)}$, $W_d^{(k)}$	weight of k^{th} encoder/decoder layer
$b_e^{(k)}$, $b_d^{(k)}$	bias of k^{th} encoder/decoder layer
$W_{f,i,C,o}^{(l)}$	weight of l^{th} LSTM layer
$b_{f,i,C,o}^{(l)}$	bias of l^{th} LSTM layer

We first introduce terms and notations that will be frequent used later, all of which are listed in Table I. Other notations will be explained along with the corresponding equations. Notice that a single LSTM cell can be regarded as a layer, in which the terms with subscript f are the parameters of forget gate, the terms with subscripts i and C are the parameters of input gate, and those with subscript o are the parameters of output gate.

1) Encoder–Decoder Architecture: Autoencoder can efficiently learn representations of data in an unsupervised way. Inspired by this, we place an encoder at the entrance of the model to capture the highly nonlinear network structure and a graph reconstructor at the end to transform the latent features back into a matrix of fixed shape. Here, however, the whole process is supervised, which is different from autoencoder, since we have labeled data (A_t) to guide the decoder to build matrices that can better fit the target distributions. In particular, the encoder, composed of multiple nonlinear perceptions, projects the high-dimensional graph data into a relatively lower dimensional vector space. Therefore, the obtained vectors could characterize the local structure of vertices in the network. This process can be characterized as

$$\begin{aligned}
\mathbf{y}_{e;i}^{(1)} &= ReLU\left(W_e^{(1)} s_i + b_e^{(1)}\right) \\
\mathbf{y}_{e;i}^{(k)} &= ReLU\left(W_e^{(k)} \mathbf{y}_{e;i}^{(k-1)} + b_e^{(k)}\right) \\
\mathbf{Y}_e^{(k)} &= \sum_{t=1}^{N-1} \mathbf{y}_{e;t}^{(k)}
\end{aligned} \tag{1}$$

where s_i donates ith graph in the input sequence S. $W_e^{(k)}$ and $b_e^{(k)}$ represent the weight and bias of the kth encoder layer, respectively. And $Y_e^{(k)}$ is the output of kth encoder layer.

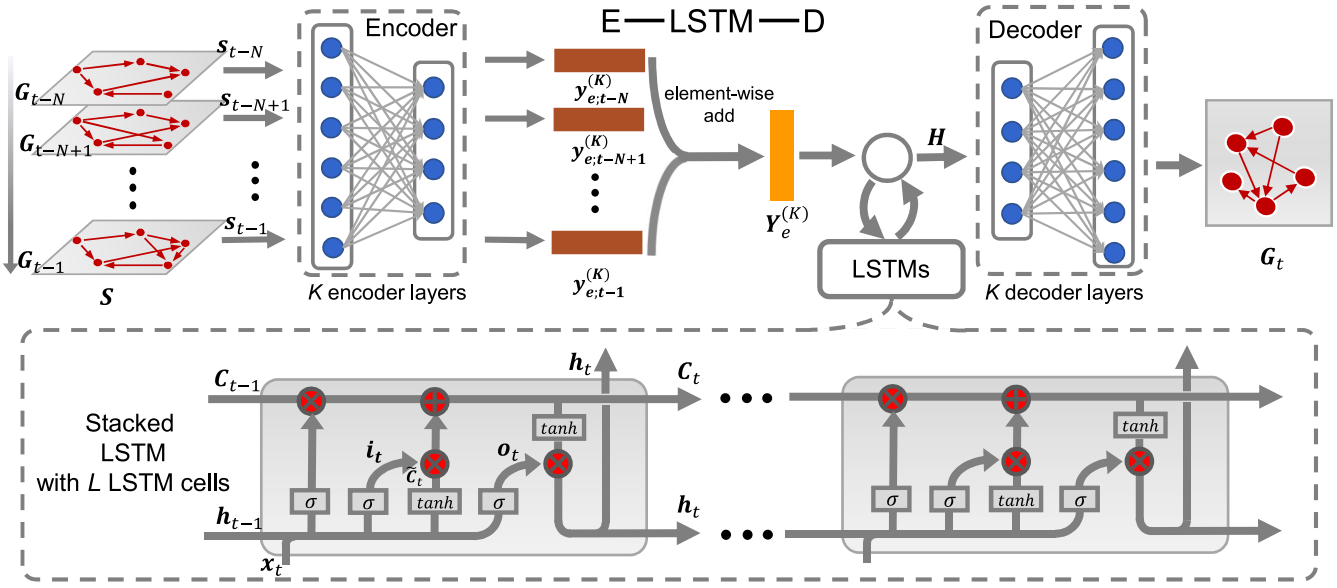


Fig. 2. Overall framework of E-LSTM-D model. Given a sequence of graphs with length N, $\{G_{t-N}, G_{t-N+1}, \ldots, G_{t-1}\}$, the encoder maps them into a lower dimensional latent space. Each graph is transformed into a matrix that represents the structural features. And then the stacked LSTM, composed of multiple LSTM cells, learns network evolution patterns from the extracted features. The decoder projects the received feature maps back to the original space to get G_t. Here, σ in LSTM cells is an activation function and we use sigmoid in this article.

For an input sequence, each encoder layer processes every term separately and then concatenates all the activations by element-wise adding. Here, we use $ReLU(x) = \max(0, x)$ [44] as the activation function for each encoder/decoder layer to accelerate convergence.

The decoder with the mirror structure of the encoder receives the latent features and maps them into the reconstruction space under the supervision of A_t, represented by

$$\begin{aligned} \mathbf{Y}_d^{(1)} &= ReLU\left(W_d^{(1)} H + b_d^{(1)}\right) \\ \mathbf{Y}_d^{(k)} &= ReLU\left(W_d^{(k)} \mathbf{Y}_d^{(k-1)} + b_d^{(k)}\right) \end{aligned} \tag{2}$$

where H is generated by the stacked LSTM and represents the features of the target snapshot rather than a sequence of features of all previous snapshots used in the encoder. $W_d^{(k)}$ and $b_d^{(k)}$ represent the weight and bias of the kth in decoder, respectively. And $\mathbf{Y}_d^{(k)}$ is the output of kth decoder layer. Another difference is the last layer of the decoder, or the output layer, uses sigmoid$(x) = [1/(1 + e^{-x})]$ as the activation function rather than $ReLU$. And the number of units of the output layer always equals to the number of nodes.

2) Stacked LSTM: Although encoder–decoder architecture could deal with the high nonlinearity, it is not able to capture the time-varying characteristics. LSTM [45], as a special kind of RNN [46], [47], can learn long-term dependencies and is introduced here to solve this problem. An LSTM consists of three gates, i.e., a forget gate, an input gate and an output gate. The first step is to decide what information is going to be thrown away from previous cell state. The operation is performed by the forget gate, which is defined as

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, Y_e^{(K)}\right] + b_f\right) \tag{3}$$

where h_{t-1} represents the output at time $t-1$ and σ donates the activation function *sigmoid*. K refers to the number of encoder layers. W_f and b_f are the weight and bias of the forget gate in LSTM, respectively. Then the input gate decides what new information should be added to the cell state. First, a sigmoid layer decides what information the input contains, i_t, should be updated. Second, a tanh layer generates a vector of candidate state values, $\tilde{C}_t$, which could be added to the cell state. The combination of i_t and $\tilde{C}_t$ represents the current memory that can be used for updating C_t. The operation is defined as

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot \left[h_{t-1}, Y_e^{(K)}\right] + b_i\right) \\ \tilde{C}_t &= \tanh\left(W_C \cdot \left[h_{t-1}, Y_e^{(K)}\right] + b_C\right) \\ C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t. \end{aligned} \tag{4}$$

W_i and W_C are the weights of the input gate in LSTM. b_i and b_C are corresponding biases. Taking the benefit of the forget gate and the input gate, LSTM cell can not only store long-term memory but also filter out the useless information. The output of LSTM cell is based on C_t and it is controlled by the output gate which decides what information, o_t, should be exported. The process is described as

$$\begin{aligned} o_t &= \sigma\left(W_o \cdot \left[h_{t-1}, Y_e^{(K)}\right] + b_o\right) \\ h_t &= o_t * \tanh(C_t) \end{aligned} \tag{5}$$

where W_o and b_o represent the weight and bias of the output gate in LSTM, respectively. A single LSTM cell is capable of learning time dependencies, but a chain-like LSTM module, namely stacked LSTM, is more suitable for processing time sequence data. Stacked LSTM consists of multiple LSTM cells that take signals as input in the order of time. We place the stacked LSTM between the encoder and the decoder to learn the patterns under which the network evolves. After receiving the features extracted at time t, the LSTM module turns them into h_t and then feed h_t back to the model at next training step. It helps the model make use of the remaining information of

previous training data. It should be always noticed that the numbers of units in encoder, LSTM cells and decoder vary when N changes. The larger N, the more units we need in the model.

The encoder at the entrance could reduce the dimension for each graph and thus keep the computation of the stacked LSTM at a reasonable cost. And the stacked LSTM which is advanced at dealing with temporal and sequential data is supplementary to the encoder in turn.

C. Balanced Training Process

L_2 distance, often applied in regression, can measure the similarity between two samples. But if we simply use it as loss function in the proposed model, the cost could probably not converge to an expected range or result in overfitting due to the sparsity of the network. There are far more zero elements than nonzero elements in A_t, making the decoder appeal to reconstruct zero elements. To address this sparsity problem, we should focus more on those existing links rather than nonexistent links in back propagation. We define a new loss function as

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a_{t;i,j} - \hat{a}_{t;i,j}\right) * p_{i,j} \\ &= \left\| \left(A_t - \hat{A}_t\right) \odot P \right\|_F^2 \end{aligned} \tag{6}$$

where $\odot$ means the Hadamard product. For each training process, $p_{i,j} = 1$ if $a_{t;i,j} = 0$ and $p_{i,j} = \beta > 1$ otherwise. And β is a positive value representing the penalty coefficient. Such penalty matrix exerts more penalty on nonzero elements so that the model could avoid overfitting to a certain extent. And we finally use the mixed loss function

$$\mathcal{L}_{\text{total}} = \mathcal{L} + \alpha \mathcal{L}_{\text{reg}} \tag{7}$$

where $\mathcal{L}_{\text{reg}}$, defined in (8), is a $\mathcal{L}_2$ regularizer to prevent the model from overfitting and α is a tradeoff parameter

$$\begin{aligned}\mathcal{L}_{\text{reg}} = &\frac{1}{2}\sum_{k=1}^{K}\left(\left\| W_e^{(k)} \right\|_F^2 + \left\| \hat{W}_d^{(k)} \right\|_F^2 \right) \\ &+ \frac{1}{2}\sum_{l=1}^{L}\left(\left\| W_f^{(l)} \right\|_F^2 + \left\| W_i^{(l)} \right\|_F^2 + \left\| W_C^{(l)} \right\|_F^2 + \left\| W_o^{(l)} \right\|_F^2 \right). \end{aligned} \tag{8}$$

The value of each element in A is either 0 or 1. The output data, however, are not one-hot encoded. They are decimals and could go to infinity or move toward the opposite direction theoretically. In order to get a valid adjacency matrix, we impose a sigmoid function at the output layer and then modify the values to 0 and 1 with 0.5 as the demarcation point. That is, there exists a link between i and j if $\hat{a}_{t;i,j} \geq 0.5$ and there is no link otherwise. To optimize the proposed model, we should first make a forward propagation to obtain the loss and then do back propagation to update all the parameters. In particular, the key operation is to calculate the partial derivative of $\partial \mathcal{L}_{\text{total}} / \partial W_{e,d}^{(k)}$ and $\partial \mathcal{L}_{\text{total}} / \partial W_{f,i,C,o}^{(l)}$.

We would like to take the calculation of $\partial \mathcal{L}_{\text{total}} / \partial W_e^{(k)}$ for instance. Taking partial derivative with respect to $W_e^{(k)}$ of (7), we have

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{total}}}{\partial W_e^{(k)}} &= \frac{\partial \mathcal{L}}{\partial W_e^{(k)}} + \alpha \frac{\partial \mathcal{L}_{\text{reg}}}{\partial W_e^{(k)}} \\ &= \frac{\partial \mathcal{L}}{\partial A_t} \cdot \frac{\partial A_t}{\partial W_e^{(k)}} + \alpha \left\| W_e^{(k)} \right\|_F. \end{aligned} \tag{9}$$

According to (6), we can easily obtain

$$\frac{\partial \mathcal{L}}{\partial A_t} = 2(A_t - \hat{A}_t) \odot P. \tag{10}$$

To calculate $\partial A_t / \partial W_e^{(k)}$, we should iteratively take partial derivative with respect to $\partial W_e^{(k)}$ on both sides of (1). After getting $\partial \mathcal{L}_{\text{total}} / \partial W_e^{(k)}$, we update the weight by

$$W_e^{(k)} = W_e^{(k)} - \lambda \frac{\partial \mathcal{L}_{\text{total}}}{\partial W_e^{(k)}} \tag{11}$$

where λ is the learning rate which is set as 1e-3 in the following experiments.

As for $\partial A_t / \partial W_d^{(k)}$ and $\partial \mathcal{L}_{\text{total}} / \partial W_{f,i,C,o}^{(l)}$, the calculation of partial derivative almost follows the same procedure. The calculation of $\partial A_t / \partial W_d^{(k)}$ goes as follows:

$$\begin{aligned} W_d^{(k)} &= W_d^{(k)} - \lambda \frac{\partial \mathcal{L}_{\text{total}}}{\partial W_d^{(k)}} \\ \frac{\partial \mathcal{L}_{\text{total}}}{\partial W_d^{(k)}} &= \frac{\partial \mathcal{L}_{\text{total}}}{\partial A_t} \cdot \frac{\partial A_t}{\partial W_d^{(k)}} + \alpha \left\| W_k^{(d)} \right\|_F \\ \frac{\partial A_t}{\partial W_d^{(k)}} &= H \cdot \prod_{i=0, i\neq k}^{K} W_d^{(i)}. \end{aligned} \tag{12}$$

It is more complicated for $\partial \mathcal{L}_{\text{total}} / \partial W_{f,i,C,o}^{(l)}$ because the recurrent network makes use of cell states at every forward propagation cycle and we need take cell states of two consecutive moments into consideration. We would take $\partial \mathcal{L}_{\text{total}} / \partial W_f^{(L)}$ as an example, which is shown below, and the rest are almost the same

$$\begin{aligned} W_f^{(L)} &= W_f^{(L)} - \lambda \frac{\partial \mathcal{L}_{\text{total}}}{\partial W_f^{(L)}} \\ \frac{\partial \mathcal{L}_{\text{total}}}{\partial W_f^{(L)}} &= \frac{\partial \mathcal{L}_{\text{total}}}{\partial A_t} \cdot \frac{\partial A_t}{\partial H} \cdot \frac{\partial H}{\partial W_f^{(L)}} + \alpha \left\| W_L^{(f)} \right\|_F \\ \frac{\partial A_t}{\partial H} &= \prod_{i=1}^{K} W_d^{(i)}. \end{aligned} \tag{13}$$

The calculation of $\partial H / \partial W_f^{(L)}$ needs to recurrently take the partial derivatives of (5). The auto-grad function implemented in deep learning framework, like TensorFlow,[2] would be useful.

D. Complexity Analysis

The complexity of E-LSTM-D almost depends on the weights of the model, just like other deep learning models. Suppose a simple E-LSTM-D model consists of an encoder

[2]https://tensorflow.google.cn/

layer with m_1 units, an LSTM layer whose hidden size is m_2 and a decoder layer, then the complexity is obtained by

$$\mathcal{O}(n) \sim n(m_1 + m_2) + 4\Big(m_1 m_2 + m_2^2 + m_2\Big) \tag{14}$$

where n is the number of nodes of G_t. The complexity changes with the structure of the model. Despite the large amount of parameters, the model could finish a single test in a few seconds with the acceleration of GPUs.

IV. Experiments

The proposed E-LSTM-D then is evaluated on five benchmark datasets, compared with four baseline methods.

A. Datasets

We perform the experiments on five real-world dynamic networks, all of which are human contact networks, where nodes denote humans and links stand for their contacts. The contacts could be face-to-face proximity, emailing, and so on. The detailed descriptions of these datasets are listed below.

1) *CONTACT [48]:* It is a human contact dynamic network of face-to-face proximity. The data are collected through the wireless devices carried by people. A link between person s (source) and t (target) emerges along with a timestamp if s gets in touch with t. The data are recorded every 20 s and multiple edges may be shown at the same time if multiple contacts are observed in a given interval.
2) *ENRON [49] and RADOSLAW [50]:* They are email networks and each node represents an employee in a mid-sized company. A link occurs every time an e-mail sent from one to another. ENRON records email interactions for nearly six months and RADOSLAW lasts for nearly nine months.
3) *FB-FORUM [51]:* The data were attained from a Facebook-like online forum of students at University of California at Irvine, in 2004. It is an online social network where nodes are users and links represent interactions (e.g., messages) between them. The records span more than five months.
4) *LKML [52]:* The data were collected from Linux kernel mailing list. The nodes represent users which are identified by their email addresses and each link donates a reply from one user to another. We only focus on the 2210 users that were recorded from 2007-01-01 to 2007-04-01 and then construct a dynamic network based on the links between these users that appeared from 2007-04-01 to 2013-12-01.

All the experiments are implemented in both long-term and short-term networks. The basic statistics of the five datasets are summarized in Table II.

Before training, we divide each dataset into pieces at a fixed interval and every ten intervals make up a time window in which the links form a snapshot. Rather than modeling the dynamic network as a sequence with incremental links, we remove the links that do not show up again in the next eight intervals with the consideration that the connections between people are probably temporary. Since the interval for different dataset may vary according to the timespan, to make sure that enough training samples could be obtained, in the experiments below, we transform each dataset into 320 snapshots with different intervals and set $N = 10$. In this case, $\{G_{t-10}, \ldots, G_{t-1}, G_t\}$ is treated as a sample with the first ten snapshots as the input and the last one as the output. The illustration is given in Fig. 3. As a result, we can get 310 samples in total. We then group the first 230 samples, with t varying from 11 to 240, as the training set, and the rest 80 samples, with t varying from 241 to 320, as the test set.

TABLE II
Basic Statistics of the Five Datasets

Dataset	$\|V\|$	$\|E_T\|$	$\bar{d}$	d_{max}	Timespan (days)
CONTACT	274	28.2K	206.2	2,092	4.0
ENRON	151	50.5K	669.8	1,841	164.5
RADOSLAW	167	82.9K	993.1	9,053	271.2
FB-FORUM	899	50.5K	669.8	5,177	164.5
LKML	2210	422.4K	34.6	47,995	2,436.3

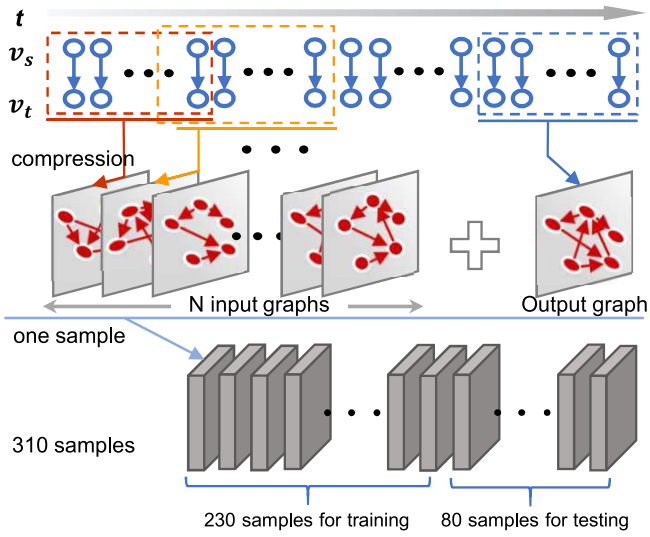


Fig. 3. Illustration of how to generate a training set and a test set.

B. Baseline Methods

To validate the effectiveness of our E-LSTM-D model, we compare it with node2vec, as a widely used baseline network embedding method, as well as four state-of-the-art DNLP methods that could handle time dependencies, including hybrid-TS [53], temporal network embedding (TNE) [54], ctRBM [38], GBDT-based temporal RBM (GTRBM) [39], and deep dynamic network embedding (DDNE) [27]. In particular, the five baselines are introduced as follows.

1) *node2vec [19]:* As a network embedding method, it maps the nodes of a network from a high-dimensional space to a lower dimensional vector space. A pair of nodes tend to be connected with a higher probability, i.e., they are more similar, if the corresponding vectors are of shorter distance.

TABLE III
PARAMETERS OF THE SIX BASELINES

Baseline	Embedding dimension	Other parameters
node2vec	80 (for contact, enron and radoslaw) 256 (for fb-forum and lkml)	p and q are grid searched over {0.5, 1, 1.5, 2}
Hybrid-TS	the same as node2vec	p_a, d_a and q_a are searched within {0, 1, 2, 3}, {0, 1} and {0, 1, 2, 3}, respectively.
TNE	(for contact, enron and radoslaw) 200 (for fb-forum and lkml)	use local auto algorithm
ctRBM	128 (for contact, enron and radoslaw) 512 (for fb-forum and lkml)	learning rate: 0.01; weight decay: 5e-4
GTRBM	the same as ctRBM	learning rate : 0.01; weight decay : 5e-4
DDNE	the same as ctRBM	learning rate : 0.01; weight decay : 5e-4

2) *Hybrid-TS [53]:* Hybrid-TS combines static link prediction approaches with time-series link prediction model to perform DNLP.
3) *TNE [54]:* It models network evolution as a Markov process and then use the matrix factorization to get the embedding vector for each node.
4) *ctRBM [38]:* It is a generative model based on temporal RBM. It first generates a vector for each node based on temporal connections and predict future linkages by integrating neighbor information.
5) *GTRBM [39]:* It takes the advantages of both tRBM and GBDT to effectively learn the hidden dynamic patterns.
6) *DDNE [27]:* Similar to autoencoder, it uses a GRU as an encoder to read historical information and decodes the concatenated embeddings of previous snapshot into future network structure.

When implementing node2vec, we set the dimension of the embedding vector as 80 for CONTACT, ENRON, and RADOSLAW which have less than 500 nodes. And for FB-FORUM and LKML with larger size, we set the dimension as 256. We grid search over {0.5, 1, 1.5, 2} to find the optimal values for hyper-parameters p and q, and then use weighted-L2 [19] to obtain the vector $e^{(u,v)}$ for each pair of nodes u and v, with each element defined as

$$e_i^{(u,v)} = |u_i - v_i|^2 \tag{15}$$

where u_i and v_i are the ith element of embedding vectors of nodes u and v, respectively. For hybrid-TS, we use node2vec described above to serve as the static link prediction approach and autoregressive integrated moving average model (ARIMA) serves as the time-series prediction model. ARIMA does prediction over link occurrence frequency series $\{x_{tij}\}$ which is defined as

$$x_{tij} = \sum\nolimits_{k \in V \text{ and } (i,k),(j,k) \in \text{ the link set of } \mathbf{S}} \log(\text{degree}(k)). \tag{16}$$

The parameters of ARIMA, the number of autoregressive terms p_a, the differential order d_a, and the number of moving average terms q_a are searched within $\{0, 1, 2, 3\}$, $\{0, 1\}$ and $\{0, 1, 2, 3\}$, respectively. For TNE, we set the dimension as 80 for CONTACT, ENRON, and RADOSLAW and 200 for FB-FORUM and LKML. We use the algorithm *local auto* to predict future links and leave other parameters as default. As for the two RBM-based methods, the parameters are mainly about the numbers of visible units and hidden units in tRBM. The number of visible units always equals to the number of corresponding network's nodes and we set the dimension of hidden layers as 128 for smaller datasets like CONTACT, ENRON, and RADOSLAW and 512 for the rest. The general settings for the six baselines are summarized in Table III. For DDNE, we set the dimension as 128 for the first three smaller datasets and 512 for the rest. When implementing our proposed model, E-LSTM-D, we choose the parameters accordingly: for the first three smaller datasets, we set $K = 1$ and $L = 2$ and add an additional layer to both encoder and decoder when for the rest two larger datasets. In the training process, we set the learning rate $\lambda = 1e-3$ and the weight decay $\alpha = 1e-4$ for all datasets. The value of K and L together with the number of units in layer are determined with the consideration of computation complexity and model's performance. Usually we set $K = 1$ for small datasets (the network with less than 500 nodes) and $K = 2$ for relatively large datasets. L is always set to 2 in this article and it can also be enlarged when it comes to larger datasets. The number of units in each layer are set accordingly. For small datasets, i.e., the number of nodes is less than 500, a single layer with 128 units is enough. And we make it larger for large datasets. Recalling that the decoder has mirror structure of the encoder, we make the number of units in the first decoder layer equal to the number of units in the last encoder layer. And the number of units in the output layer in decoder should equal to the number of nodes in the corresponding network to construction a new graph. The details of the parameters are illustrated in Table IV, which are chosen to get the best performance for each method, so as to make fair comparison.

TABLE IV
PARAMETERS OF E-LSTM-D IN THE FIVE DATASETS

Dataset	No. units in encoder	No. units in stacked LSTM	No. units in decoder
CONTACT	128	256 | 256	274
ENRON	128	256 | 256	151
RADOSLAW	128	256 | 256	167
FB-FORUM	512 | 256	384 | 384	256 | 899
LKML	1024 | 512	384 | 384	512 | 2210

C. Evaluation Metrics

There are few metrics specifically designed for the evaluation of DNLP. Usually, those evaluation metrics used in static link prediction are also employed for DNLP. The AUC is commonly used to measure the performance of a dynamic

link predictor. AUC equals to the probability that the predictor gives a higher score to a randomly chosen existing link than a randomly chosen nonexistent one. The predictor is considered more informative if its AUC value is closer to 1. Other measurements, such as precision, mean average precision (MAP), F1-score, and accuracy evaluate link prediction methods from the perspective of binary classification. All of them suffer from the sparsity problem and cannot give measurements to dynamic performances. The area under the precision-recall curve (PRAUC) [55] developed from AUC is designed to deal with the sparsity of networks. However, the removed links in the near future, as a significant aspect of DNLP, are not characterized by PR curve and thus PRAUC may lose its effectiveness in this case. Junuthula *et al.* [56] restricted the measurements to only part of node pairs and proposed the geometric mean of AUC and PRAUC (GMAUC) for the added and removed links, which can better reflect the dynamic performance. Li *et al.* [38] used SumD that counts the differences between the predicted network and the true one, evaluating link prediction methods in a more strict way. But the absolute difference could be misleading. For example, two dynamic link predictors both achieve SumD at 5. However, one predictor mispredicts five links in 10, while the other mispredicts 5 in 100. It is obvious that the latter one performs better than the former one but SumD cannot tell.

In our experiments, we choose AUC and GMAUC, and also define a new metric, *error rate*, to evaluate our E-LSTM-D model and other baseline methods.

1) *AUC:* If among n independent comparisons, there are n' times that the existing link gets a higher score than the nonexistent link and n'' times they get the same score, then we have

$$\text{AUC} = \frac{n' + 0.5n''}{n}. \tag{17}$$

Before calculation, we randomly sample nonexistent links with the same number of existing links to ease the impact of sparsity.

2) *GMAUC:* It is a metric specifically designed for measuring the performance of DNLP. It combines PRAUC and AUC by taking geometric mean of the two quantities, which is defined as

$$\text{GMAUC} = \left(\frac{\text{PRAUC}_{\text{new}} - \frac{L_A}{L_A + L_R}}{1 - \frac{L_A}{L_A + L_R}} \times 2\big(\text{AUC}_{\text{prev}} - 0.5\big) \right)^{1/2} \tag{18}$$

where L_A and L_R refer to the numbers of added and removed edges, respectively. $\text{PRAUC}_{\text{new}}$ is the PRAUC value calculated among the new links and AUC_{prev} represents the AUC for the observed links.

3) *Error Rate:* It is defined as the ratio of the number of mispredicted links, denoted by N_{false}, to the total number of truly existing links, denoted by N_{true}, which is represented by

$$\text{Error Rate} = \frac{N_{\text{false}}}{N_{\text{true}}}. \tag{19}$$

Different from SumD that only counts the absolute different links in two graphs, error rate takes the number of truly existing links into consideration to avoid deceits.

D. Experimental Results

For each epoch, we feed ten historical snapshots, $\{G_{t-10}, \ldots, G_{t-1}\}$ to E-LSTM-D and infer G_t. And it is the same for implementing the other four DNLP approaches. For the methods that are not able to deal with time dependencies, i.e., node2vec, there are following two typical treatments: 1) only using G_{t-1} to infer G_t [22] or 2) aggregating previous ten snapshots into a single network and then do link prediction [24], [27]. We choose the former one when implementing node2vec, because the relatively long sequence of historical snapshots here may carry some disturbing information that node2vec cannot handle, leading to even poor performance.

We compare our E-LSTM-D model with the five baseline methods on the performance metrics AUC, GMAUC, and error rate. Since the patterns of network evolution may change with time, the model trained by the history data may not capture the pattern in the remote future. To investigate both short-term and long-term prediction performance, we report the average values of the three performance metrics for both first 20 test samples and all the 80 samples. The results are presented in Table V, where we can see that, generally, the E-LSTM-D model outperforms the baseline methods in majority cases for both short-term and long-term prediction. Note that, our method is slightly worse than TNE on LKML when using AUC and GMAUC as the performance metric. This might be because, for a relatively large network, we need more training samples to improve the performance of E-LSTM-D. In particular, for the metrics of AUC and GMAUC, the poor performances obtained by node2vec indicate that the methods, designed for static networks, are indeed not suitable for DNLP. On the contrary, E-LSTM-D and other DNLP baselines can get much better performances, due to their dynamic nature.

Moreover, for each predicted snapshot, we also compare the predicted links with truly existing ones to obtain the error rate. We find that node2vec can easily predict much more links than the truly existing ones, leading to relatively large error rates. We argue that it might blame to the classification process that the pre-trained linear regression model is not suitable for the classification of embedding vectors. As presented in Table V, the results again demonstrate the best performance of our E-LSTM-D model on DNLP. TNE performs poorly on error rate, because it does not specially fit the distribution of the network as the other deep learning-based methods do. The dramatic difference of the error rate between E-LSTM-D and TNE indicates that this metric is a good addition to AUC to comprehensively measure the performance of DNLP. Other deep learning-based methods, like ctRBM and DDNE, have similar performances while they could not compete with E-LSTM-D in most cases. It is worth noticing that

TABLE V
DNLP PERFORMANCES ON AUC, GMAUC, AND ERROR RATE FOR THE FIRST 20 SAMPLES AND ALL THE 80 SAMPLES

Performance metric	Method	CONTACT		ENRON		RADOSLAW		FB-FORUM		LKML	
		20	80	20	80	20	80	20	80	20	80
AUC	node2vec	0.5212	0.5126	0.7659	0.6806	0.6103	0.7676	0.5142	0.5095	0.6348	0.5892
	Hybrid-TS	0.7892	0.7282	0.7221	0.6509	0.7936	0.7529	0.7604	0.7122	0.8001	0.7690
	TNE	0.9443	0.9297	0.8096	0.8314	0.8841	0.8801	**0.9810**	**0.9749**	**0.9861**	**0.9867**
	ctRBM	0.9385	0.9109	0.8468	0.8295	0.8834	0.8590	0.8728	0.8349	0.8091	0.7729
	GTRBM	0.9451	0.9327	0.8527	0.8491	0.9237	0.9104	0.9023	0.8749	0.8547	0.8329
	DDNE	0.9347	0.9433	0.7985	0.7638	0.9027	0.8974	0.9238	0.8729	0.9328	0.9115
	E-LSTM-D	**0.9908**	**0.9893**	**0.8931**	**0.8734**	**0.9814**	**0.9782**	0.9670	0.9650	0.9572	0.9553
GMAUC	node2vec	0.1805	0.1398	0.4069	0.5417	0.7241	0.7203	0.2744	0.2886	0.2309	0.2193
	Hybrid-TS	0.8299	0.7981	0.7230	0.6957	0.7408	0.6873	0.8371	0.8007	0.6829	0.6200
	TNE	0.9083	0.8958	0.8233	0.7974	0.8282	0.8251	0.9689	0.9629	**0.9839**	**0.9778**
	ctRBM	0.9126	0.8893	0.7207	0.6921	0.8004	0.7998	0.8926	0.8632	0.7723	0.7206
	GTRBM	0.9240	0.9136	0.9148	0.8675	0.9157	0.8849	0.9329	0.9117	0.6529	0.6038
	DDNE	0.8925	0.8684	0.8724	0.8476	0.8938	0.8724	0.9126	0.9023	0.7894	0.7809
	E-LSTM-D	**0.9940**	**0.9902**	**0.9077**	**0.8763**	**0.9956**	**0.9938**	**0.9926**	**0.9865**	0.8657	0.8511
Error Rate	node2vec	44.7753	25.2278	23.9053	24.8060	20.7240	21.2489	40.5109	48.5376	53.2895	61.0274
	Hybrid-TS	10.2132	12.0937	7.9902	9.2573	8.9029	8.9237	17.5391	21.0537	15.0982	19.8701
	TNE	13.1410	7.1556	23.1276	19.9167	16.7078	16.7175	19.1058	24.4350	18.5702	18.2091
	ctRBM	1.8976	1.9046	2.4890	2.7328	1.8920	2.0937	3.4509	3.6782	2.9903	3.3089
	GTRBM	1.5843	1.6953	1.5947	1.8836	1.9079	2.0031	2.2347	2.4396	2.5351	2.7942
	DDNE	1.1780	1.6036	1.7664	1.9014	1.6316	1.5941	1.9014	1.8266	2.0134	2.2258
	E-LSTM-D	**0.4011**	**0.5735**	**0.9038**	**0.9880**	**0.3392**	**0.3938**	**0.5583**	**0.5777**	**0.9840**	**1.0093**

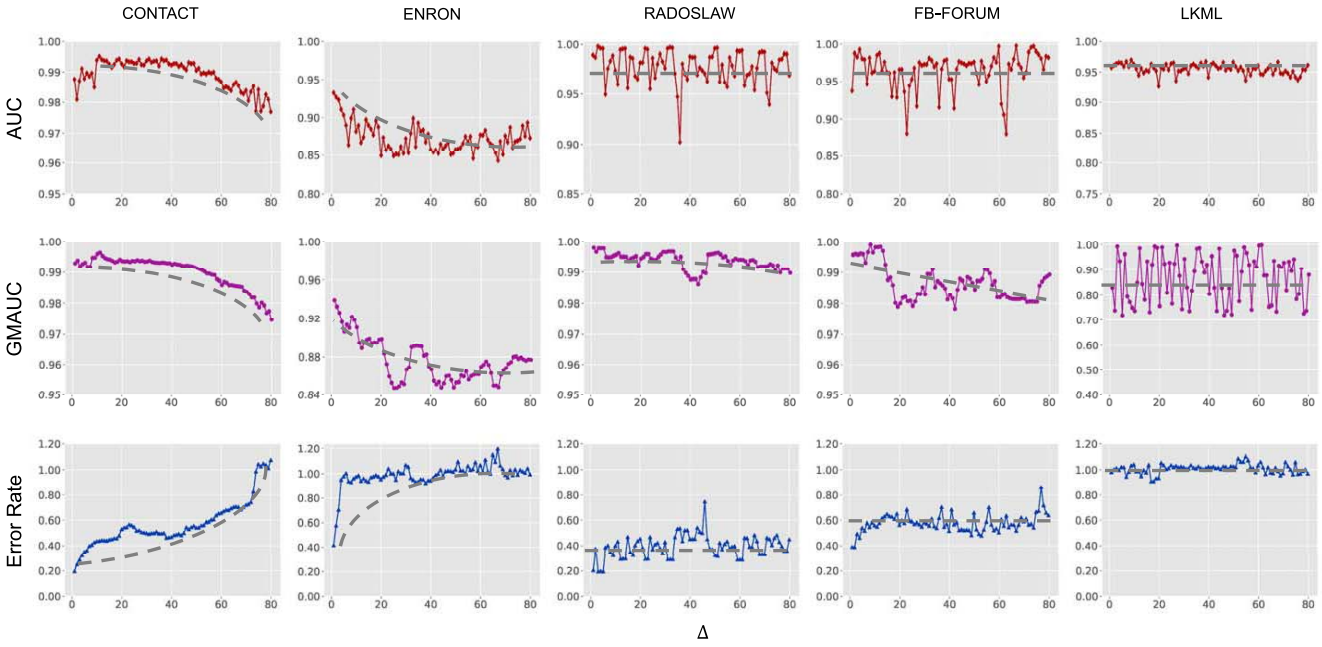


Fig. 4. DNLP performance on AUC, GMAUC, and error rate, obtained by our E-LSTM-D model, as functions of Δ for the five datasets. Δ donates the number of snapshots between the current predicted one and the 240th one. The dashed lines represent the changing tendencies.

the TNE outperforms the others on LKML from the perspective of traditional AUC and GMAUC, which shows its robustness to the scale of networks on these metrics, however, it has much larger error rate compared with the other DNLP methods.

For the 80 test samples with $G_{240+\Delta}$ as the output, where Δ varies from 1 to 80, we draw the DNLP performances on the three metrics, obtained by E-LSTM-D, as functions of Δ for the five datasets to see how long it can predict network evolution with satisfying performance. The results are shown in Fig. 4 for E-LSTM-D, where we can see that, generally, AUC and GMAUC decrease, while error rate increases, as t increases, indicating that long-term prediction on structure is indeed relatively difficult for most dynamic networks.

TABLE VI
ERROR RATE OF THE TOP 10% IMPORTANT LINKS, IN TERMS OF DC AND EBC, FOR THE FIRST 20 TEST SAMPLES AND ALL THE 80 SAMPLES

Metric for link importance	Method	CONTACT		ENRON		RADOSLAW		FB-FORUM		LKML	
		20	80	20	80	20	80	20	80	20	80
Degree centrality	node2vec	0.6279	0.6297	0.4900	0.4524	0.4735	0.5203	0.3873	0.3454	0.5034	0.5289
	Hybrid-TS	0.6529	0.7003	0.3310	0.3544	0.4218	0.4024	0.1875	0.1937	0.5532	0.5718
	TNE	0.9622	0.9551	0.3446	0.3315	0.5068	0.4413	0.0595	0.0558	0.6390	0.6288
	ctRBM	0.2739	0.3307	0.4193	0.4410	0.3028	0.3097	0.1095	0.1137	0.3291	0.3341
	GTRBM	0.2209	0.2390	0.4098	0.4322	0.2109	0.2198	0.1127	0.1239	0.2973	0.3030
	DDNE	0.1293	0.1359	0.2270	0.2133	0.0803	0.1249	0.1190	0.1088	**0.1653**	**0.1821**
	E-LSTM-D	**0.0484**	**0.1109**	**0.2182**	**0.2096**	**0.0516**	**0.0761**	**0.0160**	**0.0222**	0.1863	0.1992
Edge betweenness centrality	node2vec	0.6747	0.6509	0.4607	0.5953	0.4657	0.4397	0.6517	0.6799	0.8729	0.8698
	Hybrid-TS	0.8291	0.8506	0.7921	0.8122	0.5900	0.6467	0.5731	0.6088	0.4920	0.5355
	TNE	0.9998	0.9987	0.9598	0.9590	1.0000	1.0000	1.0000	0.9986	1.0000	0.9992
	ctRBM	0.5396	0.5619	0.6512	0.7381	0.2165	0.2291	0.4432	0.4508	0.7279	0.7503
	GTRBM	0.4418	0.4573	0.6906	0.7420	0.2399	0.2511	0.4507	0.4529	0.6370	0.6524
	DDNE	0.2713	0.2849	**0.4988**	**0.5471**	0.2083	0.2508	0.2697	0.3014	0.6435	0.6614
	E-LSTM-D	**0.2004**	**0.2547**	0.5067	0.6157	**0.1617**	**0.2159**	**0.2643**	**0.2825**	**0.5820**	**0.6126**

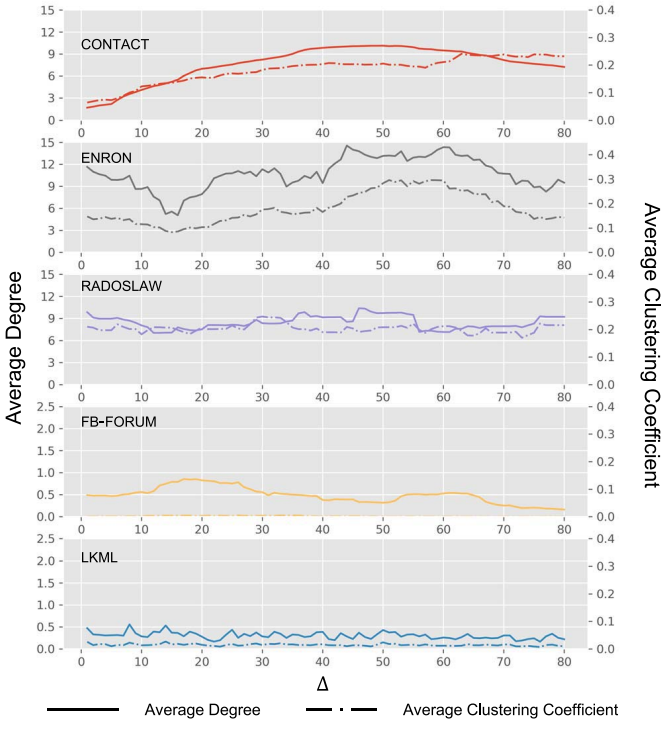


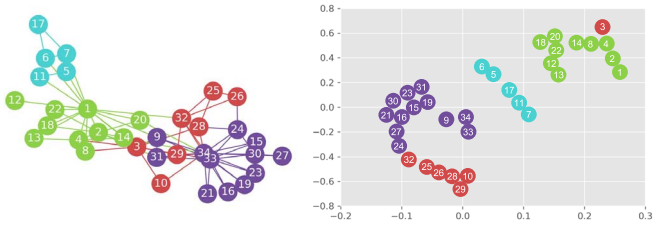
Fig. 5. Network structural properties, average degree, and average clustering coefficient, as functions of Δ for the five datasets.

Interestingly, for RADOSLAW, FB-FORUM, and LKML, the prediction performances are relatively stable, which might be because their network structures evolve periodically, making the collection of snapshots easy to predict, especially when LSTM is integrated in our deep learning framework. To further illustrate this, we investigate the changing trends of the most common structural properties, i.e., average degree and average clustering coefficient, of the five networks as Δ increases. The results are shown in Fig. 5, where we can see that these two properties change dramatically for CONTACT and ENRON, while they are relatively stable for RADOSLAW, FB-FORUM, and LKML. These results explain why we can make better long-term prediction on the last two dynamic networks.

As described above, although some methods have excellent performances on AUC, they might mispredict many links. In most real-world scenarios, however, we may only focus on the most important links. Therefore, we further evaluate our model on part of the links that are of particular significance in the network. Here, we use two metrics, degree centrality and edge betweenness centrality, to measure the importance of each link. DC is originally used to measure the importance of nodes according to the amount of neighbors. To measure the importance of a link, we use the sum of degree centralities of the two terminal nodes (source and target). We then calculate the error rate when predicting the top 10% important links. The results are presented in Table VI, which demonstrate again the outstanding performance of our E-LSTM-D model in predicting important links. It also shows that the E-LSTM-D model is more capable of learning networks' features, i.e., degree distribution and edge betweenness, which could account for the effectiveness in a way. Moreover, comparing Tables V and VI, we find that error rates on the top 10% important links are much smaller than those on all the links in the five networks by adopting any method. This indicates that, actually, those more important links are also more easily to be predicted.

E. Beyond Link Prediction

Our E-LSTM-D model learns low-dimensional representation for each node in the process of link prediction. These vectors, like those generated by other network embedding methods, contains local or global structural information that can be used in other tasks such as node classification, etc. To illustrate this, we conduct experiment on karate club dataset, with the network structure shown in Fig. 6(a). We first obtain G_{t-1} by randomly removing ten links form the original network and then use it to predict the original network



(a) (b)

Fig. 6. (a) Structure of karate club network. (b) t-SNE visualization of the embedding features obtained by our E-LSTM-D model.

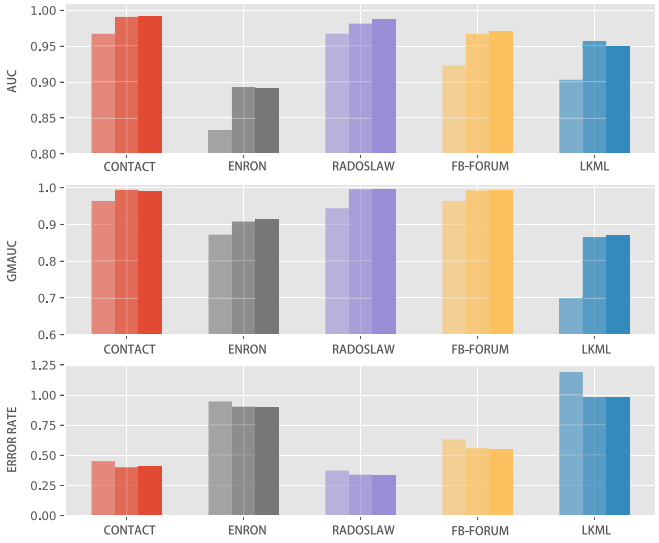


Fig. 7. Performance on the five datasets with different number of units of the first encoder layer. For each dataset, the number of the units of the first encoder layer increases at the step of 64 from left to right.

G_t. After training, we use the output of the stacked LSTM as the input to the visualization method t-SNE [57]. Besides obtaining the excellent performance on link prediction, we also visualize the embedding vectors, as shown in Fig. 6(b), where we can see that the nodes of the same class are close to each other while those of different classes are relatively far away. This indicates that the embedding vectors obtained by our E-LSTM-D model on link prediction can also be used to effectively solve the node classification problem, validating the outstanding transferability of the model.

F. Parameter Sensitivity

The performance of our E-LSTM-D model is mainly determined by three parts: 1) the structure of model; 2) the length of historical snapshots N; and 3) the penalty coefficient β. In the following, we will investigate their influences on the model performance.

1) Influence of the Model's Structure: The results shown in Table V are obtained by the models with selected structures. The numbers of units in each layer and the number of layers are set with concerns on both computation complexity and models' performance. We test the model with different number of units and encoder layers to prove the validity of the structures above. Fig. 7 shows that the performance will slightly drop with the reduction of the number of units in the first encoder layer. And further increasing the complexity has little contribution to the performance and may even lead to worse results. Table VII reports the difference of the performances between the model with an additional encoder layer which shares the same structure of the previous layer and the original model. The results show that there seems no significant improvements on AUC and GMAUC with an additional layer. But it could actually lower error rates with the increasing of the model's complexity. Overall, the general structure of E-LSTM-D can achieve state-of-the-art performance in most cases.

TABLE VII
DIFFERENCE OF THE PERFORMANCE WITH DIFFERENT NUMBER OF ENCODER LAYERS

	AUC	GMAUC	ERROR RATE
CONTACT	0.0038	0.0024	-0.1037
ENRON	0.0119	0.0206	-0.0397
RADOSLAW	0.0035	-0.0054	-0.0920
FB-FORUM	0.0029	0.0011	-0.1033
LKML	-0.0079	0.0108	-0.1375

2) Influence of Historical Snapshot Length: Usually, longer length of historical snapshots contains more information and thus may improve link prediction performance. On the other hand, snapshots from long ago, however, might have little influence on the current snapshot, while more historical snapshots will increase the computational complexity. Therefore, it is necessary to find a proper length to balance efficiency and performance. We thus vary the length of historical snapshots from 5 to 25 with a regular interval 5. The results are shown in Fig. 8(a), which tell that more historical snapshots can indeed improve the performance of our model, i.e., leading to larger AUC and GMAUC while smaller error rate. Moreover, it seems that AUC and GMAUC increase most when N changes from 1 to 10, while error rate decreases most when N changes from 1 to 20. Thereafter, for most dynamic networks, these metrics keep almost the same as N further increases. This phenomenon suggests us to choose $N = 10$ in the previous experiments.

3) Influence of the Penalty Coefficient: The penalty coefficient β is applied in the objective to avoid overfitting and accelerate convergence. When $\beta = 1$, the objective simply equals to L_2 distance. In reality, β is usually larger than 1 to help the model focus more on existing links in the training process. As shown in Fig. 8(b), we can see that the performance is relatively stable as β varies. However, for some datasets, the increasing of penalty coefficient could actually lead to slightly larger GMAUC but smaller error rate, while it has little effect on AUC. As β further increases, both GMAUC and error rate keep relatively stable. These suggest us to choose a relatively small β, i.e., $\beta \in (1, 2]$ in the experiments, varying for different datasets to obtain the optimal results.

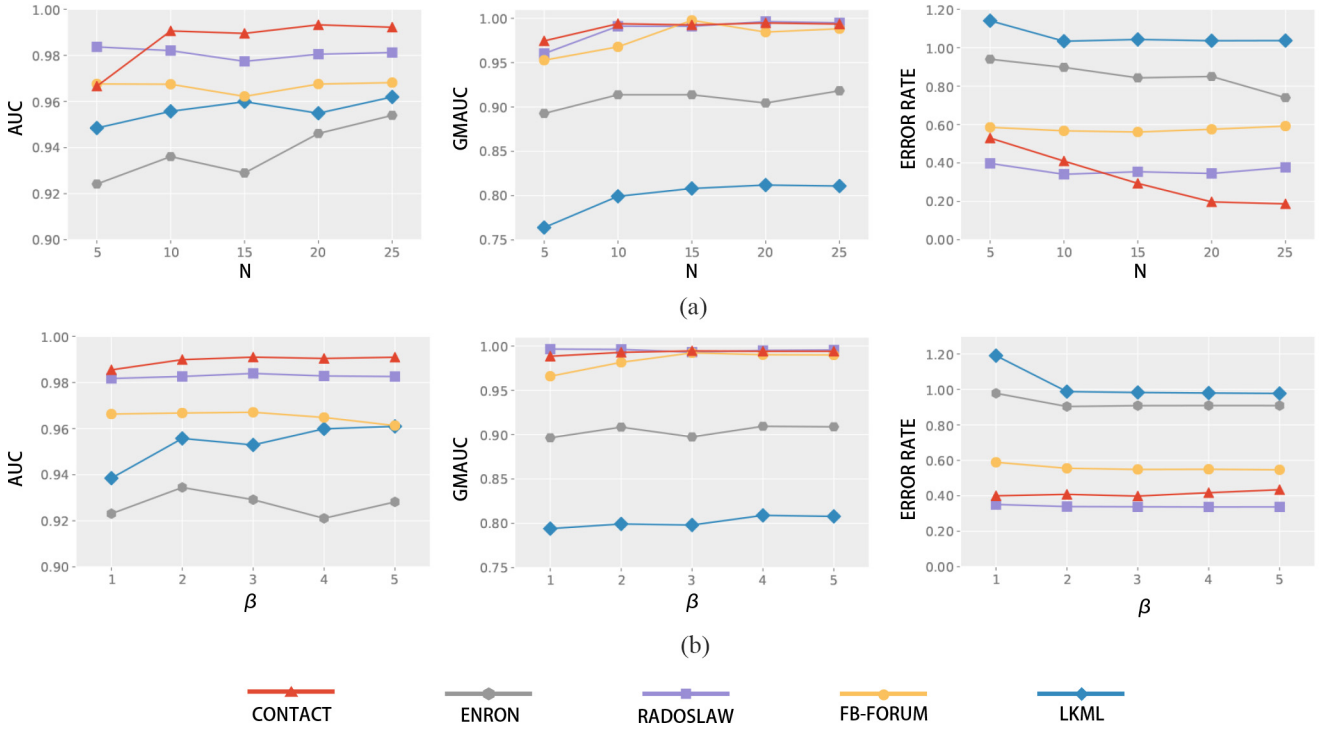


Fig. 8. Parameter sensitivity analysis of our E-LSTM-D model on five datasets. (a) Performances on AUC, GMAUC, and error rates as functions of historical snapshot length N. (b) Performances as functions of penalty coefficient β.

V. Conclusion

In this article, we propose a new deep learning model, namely E-LSTM-D, for DNLP. Specifically, to predict future links, we design an end-to-end model integrating a stacked LSTM into the architecture of encoder–decoder, which can make fully use of historical information. The proposed model learns not only the low-dimensional representations and nonlinearity, but also the time dependencies between successive network snapshots. As a result, it can better capture the patterns of network evolution. Empirically, we conduct extensive experiments to compare our model with traditional link prediction methods on a variety of datasets. The results demonstrate that our model outperforms the others and achieve the state-of-the-art performance. Moreover, when it comes to the scenarios in real life, such as analysis of protein–protein dynamic interactions and friendship prediction in social network, there is no need to compute for each link one by one any more. By using our E-LSTM-D model, we just need to feed the model with a sequence of historical snapshots and get all the results directly. It is important for those who are expertized in biology or other fields but with little knowledge of graph processing to complete DNLP task. The only thing they need to do is to transfer the temporal links into a sequence of snapshots and then the results could be obtained. Such convenience is not available when adapting other approaches. Also, in this model, besides those recent snapshots, the earlier ones could also be fully utilized, which is of significance in certain cases, e.g., communication networks.

Our future research will focus on predicting the evolution of layered dynamic networks. Besides, we will make efforts to reduce the computational complexity of our E-LSTM-D model to make it suitable for large-scale network. Also, we will study the transferability of our model on various tasks by conducting more comprehensive experiments.

References

[1] D. Ediger *et al.*, "Massive social network analysis: Mining Twitter for social good," in *Proc. IEEE 39th Int. Conf. Parallel Process. (ICPP)*, 2010, pp. 583–593.

[2] C. Fu, J. Wang, Y. Xiang, Z. Wu, L. Yu, and Q. Xuan, "Pinning control of clustered complex networks with different size," *Physica A Stat. Mech. Appl.*, vol. 479, pp. 184–192, Aug. 2017.

[3] L. Wang and J. Orchard, "Investigating the evolution of a neuroplasticity network for learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published. doi: 10.1109/TSMC.2017.2755066.

[4] J. Gao, Y. Xiao, J. Liu, W. Liang, and C. L. P. Chen, "A survey of communication/networking in smart grids," *Future Gener. Comput. Syst.*, vol. 28, no. 2, pp. 391–404, 2012.

[5] M. Kazemilari and M. A. Djauhari, "Correlation network analysis for multi-dimensional data in stocks market," *Physica A Stat. Mech. Appl.*, vol. 429, pp. 62–75, Jul. 2015.

[6] J. Sun, Y. Yang, N. N. Xiong, L. Dai, X. Peng, and J. Luo, "Complex network construction of multivariate time series using information geometry," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 1, pp. 107–122, Jan. 2019.

[7] H. Liu, X. Xu, J.-A. Lu, G. Chen, and Z. Zeng, "Optimizing pinning control of complex dynamical networks based on spectral properties of grounded Laplacian matrices," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published. doi: 10.1109/TSMC.2018.2882620.

[8] N. M. A. Ibrahim and L. Chen, "Link prediction in dynamic social networks by integrating different types of information," *Appl. Intell.*, vol. 42, no. 4, pp. 738–750, 2015.

[9] Q. Xuan, H. Fang, C. Fu, and V. Filkov, "Temporal motifs reveal collaboration patterns in online task-oriented networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 91, no. 5, 2015, Art. no. 052813.

[10] Q. Xuan, Z.-Y. Zhang, C. Fu, H.-X. Hu, and V. Filkov, "Social synchrony on complex networks," *IEEE Trans. Cybern.*, vol. 48, no. 5, pp. 1420–1431, May 2018.

[11] Q. Xuan *et al.*, "Modern food foraging patterns: Geography and cuisine choices of restaurant patrons on yelp," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 2, pp. 508–517, Jun. 2018.

[12] C. Fu *et al.*, "Link weight prediction using supervised learning methods and its application to yelp layered network," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 8, pp. 1507–1518, Aug. 2018.

[13] H. H. Lentz *et al.*, "Disease spread through animal movements: A static and temporal network analysis of pig trade in Germany," *PLoS ONE*, vol. 11, no. 5, 2016, Art. no. e0155196.

[14] A. Theocharidis, S. Van Dongen, A. J. Enright, and T. C. Freeman, "Network visualization and analysis of gene expression data using biolayout express 3D," *Nat. Protocols*, vol. 4, no. 10, pp. 1535–1550, 2009.

[15] M. E. Newman, "Clustering and preferential attachment in growing networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 64, no. 2, 2001, Art. no. 025102.

[16] T. Zhou, L. Lü, and Y.-C. Zhang, "Predicting missing links via local information," *Eur. Phys. J. B*, vol. 71, no. 4, pp. 623–630, 2009.

[17] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A Stat. Mech. Appl.*, vol. 390, no. 6, pp. 1150–1170, 2011.

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

[19] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2016, pp. 855–864.

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

[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.

[22] L. Yao, L. Wang, L. Pan, and K. Yao, "Link prediction based on common-neighbors for dynamic social network," *Procedia Comput. Sci.*, vol. 83, pp. 82–89, May 2016.

[23] N. M. Ahmed and L. Chen, "An efficient algorithm for link prediction in temporal uncertain social networks," *Inf. Sci.*, vol. 331, pp. 120–136, Feb. 2016.

[24] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Proc. 3rd Int. Workshop Learn. Represent. Big Netw. (WWW BigNet)*, 2018, pp. 969–976.

[25] Q. Xuan *et al.*, "Automatic pearl classification machine based on a multistream convolutional neural network," *IEEE Trans. Ind. Electron.*, vol. 65, no. 8, pp. 6538–6547, Aug. 2018.

[26] Q. Xuan, H. Xiao, C. Fu, and Y. Liu, "Evolving convolutional neural network and its application in fine-grained visual categorization," *IEEE Access*, vol. 6, pp. 31110–31116, 2018.

[27] T. Li, J. Zhang, P. S. Yu, Y. Zhang, and Y. Yan, "Deep dynamic network embedding for link prediction," *IEEE Access*, vol. 6, pp. 29219–29230, 2018.

[28] P. Goyal, N. Kamra, X. He, and Y. Liu, "DYNGEM: Deep embedding method for dynamic graphs," in *Proc. IJCAI Int. Workshop Represent. Learn. Graphs*, 2017.

[29] A. Ozcan and S. G. Oguducu, "Link prediction in evolving heterogeneous networks using the NARX neural networks," *Knowl. Inf. Syst.*, vol. 55, no. 2, pp. 333–360, 2018.

[30] A. Ozcan and S. G. Oguducu, "Multivariate time series link prediction for evolving heterogeneous network," *Int. J. Inf. Technol. Decis. Making*, vol. 18, no. 1, pp. 241–286, 2019.

[31] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2014.

[32] U. G. Acer, P. Drineas, and A. A. Abouzeid, "Random walks in time-graphs," in *Proc. ACM 2nd Int. Workshop Mobile Opportunistic Netw.*, 2010, pp. 93–100.

[33] N. M. Ahmed, L. Chen, Y. Wang, B. Li, Y. Li, and W. Liu, "Sampling-based algorithm for link prediction in temporal networks," *Inf. Sci.*, vol. 374, pp. 1–14, Dec. 2016.

[34] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "NetWalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2018, pp. 2672–2681.

[35] M. Brand, "Fast low-rank modifications of the thin singular value decomposition," *Linear Algebra Appl.*, vol. 415, no. 1, pp. 20–30, 2006.

[36] Z. Zhang, P. Cui, J. Pei, X. Wang, and W. Zhu, "Timers: Error-bounded SVD restart on dynamic networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 224–231.

[37] X. Ma, P. Sun, and Y. Wang, "Graph regularized nonnegative matrix factorization for temporal link prediction in dynamic networks," *Physica A Stat. Mech. Appl.*, vol. 496, pp. 121–136, Mar. 2018.

[38] 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 Min.*, 2014, pp. 289–297.

[39] T. Li, B. Wang, Y. Jiang, Y. Zhang, and Y. Yan, "Restricted Boltzmann machine-based approaches for link prediction in dynamic networks," *IEEE Access*, vol. 6, pp. 29940–29951, 2018.

[40] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. NIPS Deep Learn. Workshop*, 2014.

[41] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowl. Based Syst.*, vol. 151, pp. 78–94, Jul. 2018.

[42] Z. Zhang, J. Wen, L. Sun, Q. Deng, S. Su, and P. Yao, "Efficient incremental dynamic link prediction algorithms in social network," *Knowl. Based Syst.*, vol. 132, pp. 226–235, Sep. 2017.

[43] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic network embedding by modelling triadic closure process," in *Proc. AAAI*, 2018.

[44] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," in *Proc. ICML Deep Learn.*, 2015, pp. 1–5.

[45] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," in *Proc. 9th Int. Conf. Artif. Neural Netw. (ICANN)*, 1999, pp. 850–855.

[46] Z. C. Lipton, J. Berkowitz, and C. Elkan. (2015). *A Critical Review of Recurrent Neural Networks for Sequence Learning*. [Online]. Available: https://arxiv.org/abs/1506.00019

[47] Z. Han *et al.*, "SeqViews2SeqLabels: Learning 3D global features via aggregating sequential views by RNN with attention," *IEEE Trans. Image Process.*, vol. 28, no. 2, pp. 658–672, Feb. 2019.

[48] (Apr. 2017). *Haggle Network Dataset—KONECT*. [Online]. Available: http://konect.uni-koblenz.de/networks/contact

[49] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proc. 29th Conf. Artif. Intell. (AAAI)*, 2015, pp. 4292–4293. [Online]. Available: http://networkrepository.com

[50] R. Michalski, S. Palus, and P. Kazienko, *Matching Organizational Structure and Social Network Extracted From Email Communication* (Lecture Notes in Business Information Processing) vol. 87. Berlin, Germany: Springer, 2011, pp. 197–206.

[51] (Apr. 2017). *Facebook Wall Posts Network Dataset—KONECT*. [Online]. Available: http://konect.uni-koblenz.de/networks/facebook-wosn-wall

[52] (Apr. 2017). *Linux Kernel Mailing List Replies Network Dataset—KONECT*. [Online]. Available: http://konect.uni-koblenz.de/networks/lkml-reply

[53] Z. Huang and D. K. J. Lin, "The time-series link prediction problem with applications in communication surveillance," *INFORMS J. Comput.*, vol. 21, no. 2, pp. 286–303, 2009.

[54] L. Zhu, D. Guo, J. Yin, G. Ver Steeg, and A. Galstyan, "Scalable temporal latent space inference for link prediction in dynamic social networks," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 10, pp. 2765–2777, Oct. 2016.

[55] Y. Yang, R. N. Lichtenwalter, and N. V. Chawla, "Evaluating link prediction methods," *Knowl. Inf. Syst.*, vol. 45, no. 3, pp. 751–782, 2015.

[56] R. R. Junuthula, K. S. Xu, and V. K. Devabhaktuni, "Evaluating link prediction accuracy in dynamic networks with added and removed edges," in *Proc. IEEE Int. Conf. Big Data Cloud Comput. (BDCloud) Soc. Comput. Netw. (SocialCom) Sustain. Comput. Commun. (SustainCom)(BDCloud-SocialCom-SustainCom)*, 2016, pp. 377–384.

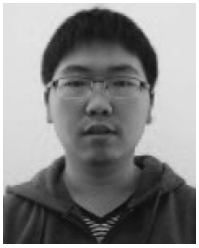
[57] L. Maaten and G. Hinton, "Visualizing data using T-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

Jinyin Chen received the B.S. and Ph.D. degrees in control theory and engineering from the Zhejiang University of Technology, Hangzhou, China, in 2004 and 2009, respectively.

She is currently an Associate Professor with the College of Information Engineering, Zhejiang University of Technology. Her current research interests include intelligent computing, social network data mining, optimization, and network security.

Jian Zhang received the B.S. degree in automation from the Zhejiang University of Technology, Hangzhou, China, in 2017, where he is currently pursuing the M.S. degree in control theory and engineering with the College of Information and Engineering.

His current research interests include network analysis and deep learning, especially the intersection of the two fields.

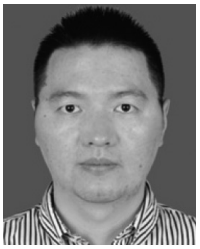


Xuanheng Xu received the bachelor's degree in automation from the Zhejiang University of Technology, Hangzhou, China, in 2016, where he is currently pursuing the master's degree in control theory and engineering with the College of Information Engineering.

His current research interests include data mining and applications, industrial safety, and artificial intelligence.

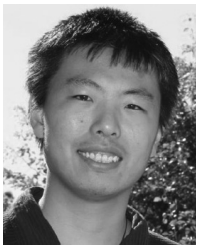
Chenbo Fu received the B.S. degree in physics from the Zhejiang University of Technology, Hangzhou, China, in 2007, and the M.S. and Ph.D. degrees in physics from Zhejiang University, Hangzhou, in 2009 and 2013, respectively.

He was a Post-Doctoral Researcher with the College of Information Engineering, Zhejiang University of Technology, and a Research Assistant with the Department of Computer Science, University of California at Davis, Davis, CA, USA, in 2014. He is currently a Lecturer with the College of Information Engineering, Zhejiang University of Technology. His current research interests include network-based algorithm design, social network data mining, chaos synchronization, network dynamics, and machine learning.



Dan Zhang received the Ph.D. degree in control theory and control engineering from the Zhejiang University of Technology, Hangzhou, China, in 2013.

He is currently an Associate Professor with the College of Information Engineering, Zhejiang University of Technology. His current research interests include hybrid systems, cyber-physical systems, artificial intelligence, and machine learning.



Qingpeng Zhang (M'10) received the B.S. degree in automation from the Huazhong University of Science and Technology, Wuhan, China, in 2009, and the Ph.D. degree in systems and industrial engineering from the University of Arizona, Tucson, AZ, USA, in 2012.

He was a Post-Doctoral Research Associate with the Department of Computer Science, Tetherless World Constellation, Rensselaer Polytechnic Institute, Troy, NY, USA. He is currently an Assistant Professor with the School of Data Science, City University of Hong Kong, Hong Kong. His current research interests include social computing, complex networks, data mining, and semantic Web.

Dr. Zhang is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SCIENCE and IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.

Qi Xuan (M'18) received the B.S. and Ph.D. degrees in control theory and engineering from Zhejiang University, Hangzhou, China, in 2003 and 2008, respectively.

He was a Post-Doctoral Researcher with the Department of Information Science and Electronic Engineering, Zhejiang University, from 2008 to 2010, and a Research Assistant with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, in 2010 and 2017. From 2012 to 2014, he was a Post-Doctoral Fellow with the Department of Computer Science, University of California at Davis, CA, USA. He is a member of IEEE and is currently a Professor with the Institute of Cyberspace Security, College of Information Engineering, Zhejiang University of Technology, Hangzhou. His current research interests include network science, graph data mining, cyberspace security, machine learning, and computer vision.