Dynamic Network Link Prediction by Learning Effective Subgraphs using CNN-LSTM

Kalyani Selvarajah
School of Computer Science
University of Windsor
Windsor, ON, Canada
selva111@uwindsor.ca

Kumaran Ragunathan
School of Computer Science
University of Windsor
Windsor, ON, Canada
ragunat@uwindsor.ca

Ziad Kobti
School of Computer Science
University of Windsor
Windsor, ON, Canada
kobti@uwindsor.ca

Mehdi Kargar
Ted Rogers School of Management
Ryerson University
Toronto, ON, Canada
kargar@ryerson.ca

***Abstract*—Predicting the future link between nodes is a significant problem in social network analysis, known as Link Prediction (LP). Recently, dynamic network link prediction has attracted many researchers due to its valuable real-world applications. However, most methods fail to perform satisfying prediction accuracy in various types of networks because the dynamic LP in evolving networks is struggling with spatial and nonlinear transitional patterns. Besides this, existing methods mostly involve the whole network and target link for the LP process. It leads to high computational costs. This paper aims to address these issues by proposing a novel framework named DLP-LES using deep learning methods. DLP-LES uses common neighbors based subgraph of a target link and learns the transitional pattern of it for a given dynamic network. We extract a set of heuristic features of the evolving subgraph to gather additional information about the target link. In this way, we avoid examining the entire network. Additionally, our model introduces new mechanisms to reduce computational costs. DLP-LES generates a lookup table to keep the required information of links of the network and uses a hashing method to store and fetch link information. We propose an algorithm to construct feature matrices of the evolving subgraph to learn transitional link patterns. Our model transforms the dynamic link prediction to a video classification problem, and uses Convolutional Neural Networks with Long Short-Term Memory neural networks. To verify the effectiveness of DLP-LES, extensive experiments are carried out on five real-world dynamic networks. We compare those results against four network embedding methods and basic heuristic methods.**

***Index Terms*—Social Networks, Dynamic Networks, Link Predictions, Subgraphs**

I. Introduction

Dynamic network analysis has become an important research problem in recent years because it resembles the evolving nature of real-world networks. It has taken a great deal of attention from various fields, including social science [1], economics [2], and biology [3]. Dynamic networks evolve over time, and nodes and links may appear or disappear as time goes by. One of the primary areas of research in dynamic networks is temporal link prediction, which attempts to predict the links in the future using the transformation of a sequence of networks. LP has several applications including friend recommendation [4], classify the behavior and motion of people [5], and disease gene prediction [6].

Numerous studies have been performed in a static network setting, which considers a single snapshot of a network at time t and is used to determine new links in time $t'(> t)$. Simple heuristic methods, often based on topological properties of the network, such as common neighbors [7], Adamic-Adar [4] and Katz [8] or a combination of such heuristics are well-defined for static networks. Link prediction in a dynamic network is a challenging and complex process. It has a completely new dimension of analysis because the history of network evolution provides more information to detect potential or future links. The dynamic network settings can be generally formulated as the sequence of network snapshots, as shown in Figure 1, where the behavior of each snapshot can be described as a static network at a time. To deal with dynamic network link prediction, various methods have been proposed in the literature [9]–[12]. These methods include network embedding techniques such as DeepWalk [13], LINE [14] and Node2Vec [12] and deep learning techniques [9]–[11]. The approach in [15] and [16] have explored the usage of heuristic methods in the dynamic network link prediction.

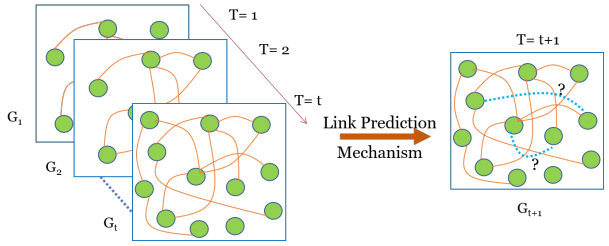


Figure 1. The representation of a dynamic network G with a series of snapshots from time 1 to t as a input and a snapshot at time $t + 1$ as a output

Most of the existing approaches in both static and dynamic settings focused only on the target nodes, source and destination of the link and entire network for the prediction. However, the target nodes and their neighbor nodes play a high impact on link prediction, and analyzing the portion of the whole network reduce time complexity. Recent groundbreaking methods in static networks, WLMN [17] and SEAL [18] proposed neural network approaches to automate the selection of best heuristic for a given network, and introduced subgraph extraction methods, based on neighbor nodes, of the target links for the prediction. However, PLACN [19] claimed that subgraphs by common neighbor nodes of target link have

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

additional information than the subgraph from just neighbor nodes, and achieved outstanding results in various types of static networks. Motivated from this, we extract subgraph from common neighbors of target links and extend the benefits of heuristics to the dynamic network settings. We believe that considering subgraph based on common neighbors of a target link bring a huge advantage to analyze the evolving pattern of the target link in the dynamic network.

To the best of our knowledge, we are the first people using the common neighbor based subgraph for the dynamic network link prediction problems. Our proposed model, DLP-LES, begins with extraction of common neighbor based subgraph from the last snapshot of a given dynamic network, and analyzes the transitional patterns of subgraph using heuristic features throughout each time step. Due to complexity, most of the research in dynamic settings ignored the link weight, and only considered the existence and absence of the link. In DLP-LES, we include link weight as an additional information with heuristic features. Besides this, we construct a lookup table with every information of links of a given dynamic network. The primary purpose of the lookup table is to reduce the time and space complexity when we frequently use the information of the same links. We elaborate this further in the section for constructing the lookup table. Thereafter, this study introduces an efficient encoding method to label the subgraph's nodes. It is another significant task in our model to maintain the consistency of the subgraph when we train the neural networks. We believe that examining the evolving heuristic features of the subgraph has a significant impact on introducing a new link between any two nodes of a dynamic network.

We summarize our main contributions as follows:

- We introduce a method to generate a lookup table to keep the record of links' information of a given dynamic network, and use a hashing method to fetch essential information when required.
- We introduce a novel encoding method for subgraph labeling.
- We propose an algorithm to construct feature matrices for the subgraph efficiently.
- We propose a new framework, DLP-LES, for the dynamic network link prediction using Convolutional Neural Networks (CNN) to extract higher-level features of subgraph efficiently and Long Short-Term Memory (LSTM) neural networks to learn long-range dependencies of sequential data and capture the evolving patterns of the subgraph in the dynamic networks.

II. Related Works

Link Prediction in Dynamic Networks (DN) is one of the hot topics in social network analysis. Modeling this problem is a complex and highly challenging process. Diverse methods have been proposed in the literature to improve the accuracy of the predictions.

Heuristic methods such as common neighbors [7], Adamic-Adar [4] and Katz [8] consider the topological structure to predict the links in the future, which are very famous for static networks. Yao et al. [20] proposed a modified common neighbors formula and use of time-decay to handle DN. Some others [15], [16] extended the application of these heuristics to DN settings. Chiu et al. [16] proposed a weak estimator to decide the link existence based on a random probability function, while Kaya et al. [15] explored aggregate heuristic metrics by weighting snapshots.

Besides heuristic based prediction methods, various machine learning techniques have been applied for LP in DN. Gao et al. [21] performed a method by combining the latent matrix factorization method and graph regularization technique to learn the structural information of time evolving patterns of links. Yu et al. [22] proposed a model (LINE) with spatial and temporal consistency to tackle DN prediction. They represented the network structure as a function of time. Ma et al. [23] proposed a non-negative matrix factorization (NMF) framework which incorporated the dynamic information of historical snapshots by using the graph regularization technique.

In addition to the above two methods, deep learning approaches have become cutting-edge techniques in DN link predictions. Li et al. [9] proposed a framework using boltzmann machine which predicts links based on individual transition variance in addition to influence introduced by local neighbors. The authors of [24] proposed a network embedding method to handle DN settings. They incorporated both the internal and dynamic transition structures in their design. Lei et al. [25] proposed a model using GCN, LSTM, and GAN to solve the challenges in temporal LP. They used graph convolutional network (GCN) to study the local topological structure, LSTM to analyze the evolving features of networks, and generative adversarial networks (GAN) to handle weighted DN.

Some other methods are also used in temporal network LP. CA Bliss et al. [26] employed evolutionary algorithms to predict the links on DN by applying the Covariance Matrix Adaptation Evolution Strategy.

III. Problem Definition

A dynamic network can be defined as a sequence of network snapshots considered within a specific time interval where as a static network does not change the topological structure over time. In this paper, we consider an undirected, weighted dynamic network.

Given a series of snapshots $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_t\}$ of an evolving graph $\mathcal{G}$, where $\mathcal{G}_p = \langle \mathcal{V}, \mathcal{E}_p \rangle$ represents a snapshot of the given dynamic network at time p. In this study, $\mathcal{V}$ specifies the same vertices shared by all snapshots. $\mathcal{E}_p$ specifies the links or edges of the snapshot at time p. A snapshot $\mathcal{G}_p$ can be treated as a static network, and can be written as an adjacency matrix $A_p = [a_p(i,j)]_{|V|\times|V|}$ to represents the corresponding static topological structure, where $a_p(i,j) > 0$ if the vertices $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$ are connected, otherwise, $a_p(i,j) = 0$. The sequence of graphs $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_t\}$ correspond to a list of symmetric adjacency matrices $\{A_1, A_2, \ldots, A_t\}$.

Definition 1. ***(Link Prediction in Dynamic Networks)*** *Given that a sequence of snapshots with length k have the corresponding adjacency matrices $\{A_{t-k}, A_{t-k+1}, \ldots, A_t\}$, the primary objective of link prediction in dynamic networks is to model a framework to learn the following function to predict the topological changes, mainly in links at time $t+1$.*

$$A_{(t+1)} = f(A_{t-k}, A_{t-k+1}, , \ldots, A_t) \tag{1}$$

where $f(A_{t-k}, A_{t-k+1}, \ldots, A_t)$ represents the model required to predict the adjacency matrix $A_{(t+1)}$ at time $t+1$.

IV. Modeling Dynamic Networks Link Prediction

In this section, we discuss the backgrounds of required theories and techniques to model a framework for predicting links in the future.

A. Heuristic Methods

Several heuristics have been proposed extensively to solve link prediction problems on static networks, such as Common neighbors, Adamic-Adar and Katz. We can categorize them as first, second, and high order heuristics based on their complexity to perform. The first and second order heuristics are efficiently computable, and measure diverse aspects of the network topology such as closeness and similarity between any two nodes in the social networks. The following section lists down five such heuristics used in this paper, where $\Gamma(v)$ and $\Gamma(u)$ specify the set of neighbors for nodes v and u respectively.

1) Common Neighbors (CN): The idea of CN is that if the nodes share links with other nodes, the chances of forming a new link is high. It is the most simplest method and counts the number of neighbors that any two vertices v and u directly interact with.

$$\mathcal{CN} = |\Gamma(v) \cap \Gamma(u)| \tag{2}$$

2) Jaccard Coefficient (JC): The CN measures the relative similarities between any two nodes because it does not consider the proportion of links shared; it is not normalized. JC produces the normalized form of CN based on the total number of neighbors both v and u have.

$$\mathcal{JC} = \frac{|\Gamma(v) \cap \Gamma(u)|}{|\Gamma(v) \cup \Gamma(u)|} \tag{3}$$

3) Adamic-Adar (AA): The AA is the modified version of JC. The primary purpose of AA is to give a higher priority to the common neighbors with very few neighbors or lower degree.

$$\mathcal{AA} = \sum_{k \in |\Gamma(v) \cup \Gamma(u)|} \frac{1}{log|\Gamma(k)|} \tag{4}$$

4) Preferential Attachment (PA): The concept of PA is if a node has a higher degree, the chances of making new connections is high.

$$\mathcal{PA} = |\Gamma(v).\Gamma(u)| \tag{5}$$

5) Resource Allocation (RA): RA metric is much more similar to AA. The difference is that RA gives higher priority to low-degree common neighbors than AA.

$$\mathcal{RA} = \sum_{k \in |\Gamma(i) \cup \Gamma(j)|} \frac{1}{|\Gamma(k)|} \tag{6}$$

B. CNN-LSTM

Here, we briefly introduce the components of a CNN-LSTM, and it's significance in our model. DLP-LES comprises Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) neural networks [27]. CNN has been proved to be successful in image-related tasks including, image classification, object detection, and computer vision. Here, we take the advantage of CNN model in extracting heuristic features of the subgraph's adjacency matrices, which can be treated as an image in DLP-LES. In other words, we can transform the input data into an image to use in CNN. The LSTM model has been proved to be extremely effective in capturing long-term temporal correlations with arbitrary length. It can be used in several other applications, including text classification, handwriting recognition and speech recognition. The LSTM model preserves long-term dependencies effectively using three different gates:input gate activation (i_t), output gate activation (o_t) and forget gate (f_t). Its unit has a memory (c_t) cell, and its neuron input and output are x_t and h_t respectively at time step t.

$$LSTM = \begin{cases} i_t : \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\ f_t : \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\ o_t : \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \\ g_t : \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \\ c_t : f_t \odot c_{t-1} + i_t \odot g_t \\ h_t : o_t \odot \tanh(c_t) \end{cases} \tag{7}$$

where σ specifies the sigmoid activation function, bs denotes the bias, and Ws specify weight.

V. Architecture of DLP-LES Model

In this section, we describe our DLP-LES framework for link prediction. The proposed model has the following four major steps:

1) Link features lookup table construction.
2) Subgraph extraction and labeling.
3) Features matrix construction for links in subgraphs.
4) Modeling with CNN and LSTM.

At the beginning, we have a dynamic network $\mathcal{G}$ with the information of source and destination nodes that are observed within a time stamp T. Before we use this network, it needs to be arranged as a series of snapshots with equal time intervals $\Delta(t)$. In our case, $\{\mathcal{G}t_k, \mathcal{G}_{t-k+1}, \ldots, \mathcal{G}_{t-1}\}$ is treated as a sample first k snapshots with equal intervals as the input and the last $(t)^{th}$ snapshot as the output.

We name our proposed framework DLP-LES (**D**ynamic network **L**ink **P**rediction by **L**earning **E**ffective **S**ubgraphs),

to highlight our focus on efficient common neighbor based subgraph to handle dynamic link prediction.

A. Link Features Lookup Table Construction

In this framework, features of links play a significant role to predict links in the future. As described in the above section, we have the sequence of snapshots of a given dynamic network $\mathcal{G}$. For the last snapshot $\mathcal{G}_t$, we extract subgraphs of the targeted links for prediction and analyze the heuristic features of each link in the subgraph. Evaluating heuristic features of links might be a repetitive process if we consider two targeted links from the subgraphs which have common links.

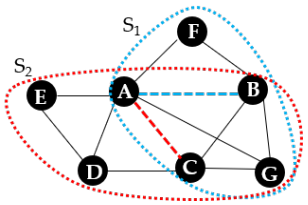


Figure 2. The representation of two subgraphs with common links for different targeted links. Subgraph S_1 with blue line is for the target link AB while Subgraph S_2 with red line is for the target link AC.

For example, consider two subgraphs S_1 and S_2 for the targeted links AB and AC as shown in figure 2. In our case, we need to calculate heuristic features for S_1 of links $\{AB, AF, FB, AC, AG, CG, BC, BG\}$ and S_2 of links $\{AB, AE, AD, AC, AG, ED, DC$
$, CG, BG, BC\}$. We need to repeat the calculation of feature for the common links $\{AB, AC, AG, CG, BC, BG\}$.

To avoid this repeated process, we initially build a lookup table $\langle\mathcal{R}\rangle$ to store the following information for every links of a given network G.

$$\langle v,u\rangle = \begin{cases} \text{Minimum Number of Hops} \\ \text{Average Path Weight} \\ \text{Time Stamp:} \begin{cases} [t_k\langle cn, jc, aa, pa, ra, w\rangle] \\ [t_{k-1}\langle cn, jc, aa, pa, ra, w\rangle] \\ . \\ . \\ [t_t\langle cn, jc, aa, pa, ra, w\rangle] \\ [t_{t+1}\langle cn, jc, aa, pa, ra, w\rangle] \end{cases} \end{cases}$$

where $t_k\langle cn, jc, aa, pa, ra, w\rangle$, specifies the heuristic feature values of snapshots at t_k, and $\langle k, (k-1), \ldots, t, (t+1)\rangle$ specifies the series of time. The minimum number of hops represents the number of minimum hops between v and u from the last snapshot $t+1$, and the average path weight is the ratio between path weight of minimum hop and number of minimum hops from the last snapshot $t+1$. The average path weight of a link can be calculated as below,

$$w_{avg}\langle v,u\rangle = \frac{1}{2}\left(\frac{1}{h}\sum_{p=0}^{h} w_p\right) \tag{8}$$

where w_p specifies the shortest path distance between v and p, add up to node u and h represents the number of hops between v and u.

Our primary objective is to construct a repository $\langle\mathcal{R}\rangle$ that can be used to retrieve information of links without calculating it repeatedly. Rather than accessing the repository as a table, using a hashing function to access the information has more benefits. For this purpose, we formulate the following hashing function.

$$f(\langle v,u\rangle|\langle u,v\rangle) = \langle\mathcal{R}\rangle \tag{9}$$

where v and u are any vertices and the hashing function can provide the feature information for any order of vertex pair. We first convert the node pair $\langle v,u\rangle$ to a unique key, which is the same key for the nodes v and u in any order (ie $\langle v,u\rangle$, $\langle u,v\rangle$). To collect any information that we store in a lookup table, we can use the above function 9. So, the complexity is O(1) to gather information of a given link.

B. Subgraph Extraction and Node Labeling

Another primary process of our model is subgraph extraction. Although few subgraph extraction methods are proposed in the existing literature [17], [18], the extracted subgraphs using existing methods for LP do not have sufficient information. We use common neighbors of any targeted nodes v and u to create subgraphs. The common neighbors can be collected from different hops of both nodes, v and u. Rather than collecting just neighbor nodes, collecting common neighbors of both nodes v and u will have more information to decide the existence of link between them in the future. We set a threshold value Θ to keep the number of nodes limit in the subgraph.

Definition 2. ***(Subgraph based on Common neighbors)*** *For a dynamic network of last snapshot $\mathcal{G}_t = \langle\mathcal{V}, \mathcal{E}_t\rangle$, $\mathcal{G}' = \langle\mathcal{V}', \mathcal{E}'_t\rangle$ is a subset of sets of common neighbor nodes of two nodes $v_i \in \mathcal{V}'$ and $v_j \in \mathcal{V}'$, and are denoted as $\Gamma(v_i)$ and $\Gamma(v_j)$ if and only if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}'_t \subseteq \mathcal{E}_t$, $\mathcal{V}'$ is a set of common neighbors for the targeted links, and $|\mathcal{V}'| = \Theta$.*

The algorithm 1 shows the step by step procedure to extract subgraph $\mathcal{G}'$ for a given link between v and u from a dynamic network of last snapshot $\mathcal{G}_t$. Since dynamic networks evolve over time, most recent snapshot has more reliable information for the link predictions in the future [28]. At the beginning, the first order common neighbors $\Gamma^1(i) \cap \Gamma^1(j)$ of v and u are collected and stored to a node list N_Θ. Then, gradually increase the order of common neighbors $(\Gamma^2(i) \cap \Gamma^2(j)), (\Gamma^3(i) \cap \Gamma^3(j)), ...$, until $|N_\Theta| \geq \Theta$, where $\Gamma^p(q)$ is the p^{th} order neighbor nodes of node q.

The above procedure may return the number of node list of the subgraph more than the defined threshold limit, $|N_\Theta| > \Theta$. At this point, each extracted subgraphs of last snapshot of a given dynamic network may have different number of node list. This inconsistency situation creates problem when we train convolutional neural network. We solve this issue by

Algorithm 1 Common Neighbor Based Subgraph Extraction

Input: Target link E_{vu}, a snapshot graph $\mathcal{G}_p = \langle(\mathcal{V}, \mathcal{E}_p)$ at time p.
Output: Subgraph $\langle G' \rangle$ for the link E_{vu}.

$N_\Theta = \{v, u\}$
$N_{temp} = \{\}$
$h = 1 \leftarrow$ number of order
while $|N_\Theta| \leq \Theta$ **do**
  $N_{temp} = \Gamma^h(v) \cap \Gamma^h(u)$
  $N_\Theta = N_\Theta \cup N_{temp}$
  $h \leftarrow h + 1$
end while
$\langle G' \rangle \leftarrow$ subgraph $G(N_\Theta)$
return $\langle G' \rangle$

removing some nodes when we process labeling to keep the number of nodes limits equivalent to Θ.

Node labeling is another significant process in this study. It helps to maintain the consistency of the subgraphs. After we extract the subgraph, the nodes containing the target link get the labels 1 and 2. We use the node list N_Θ, which returns from subgraph extraction. We then remove the nodes belonging to the targeted link from N_Θ. To order the remaining nodes $R_\Theta = N_\Theta - \{1, 2\}$, we use the information of average minimum hops and average path weight. We can use the hashing function equation 9 to get the required information. However, we need an average number of hops (H_{Avg}) and average path weight (W_{Avg}). So, we use the following formulas to evaluate H_{Avg} and W_{Avg}.

$$H_{avg}\langle v, u \rangle = \frac{1}{2}(h_{v,i} + h_{i,u}) \tag{10}$$

$$W_{avg}\langle v, u \rangle = \frac{1}{2}(w_{avg}\langle v, i \rangle + w_{avg}\langle i, u \rangle) \tag{11}$$

where $h_{v,i}$ (resp. $h_{i,u}$) is the minimum number of hops between v and i (resp. $h_{i,u}$), and $w_{avg}\langle v, i \rangle$ (resp. $w_{avg}\langle i, u \rangle$) is the average path weight of the link $\langle vi \rangle$ (resp. $\langle iu \rangle$).

Our aim is to order the nodes in R_Θ in a consistent way. We can sort them first with average hop in ascending order and then with average path weight in descending order which helps to break tie from first ordering. Therefore, we come up with an idea to encode both H_{avg} and W_{avg} into single form which reduces the complexity. For example, to generate a encoder to the node with $H_{avg} = 1.5$ and $W_{avg} = 1.75$, we encode as shown in figure 3, where the first portion indicates the value of $H_{avg} = 1.5$ and last portion indicates the reciprocal value of $W_{avg} = 1.75$, which remains always within the range $0 < 1/W_{avg} \leq 1$ in our case.

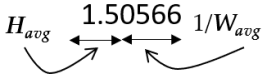
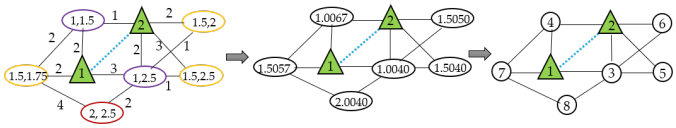


Figure 3. Encoding Format Example: first portion of the encoder specifies the average hop (H_{avg}) and the last portion specifies the reciprocal of average path weight (W_{Avg}).

Algorithm 2 Subgraph Node Labeling

Input: Nodes List N_Θ, Target link E_{vu}, Subgraph $\langle \mathcal{G}' \rangle$
Output: Ordered nodes list O_Θ

$O_\Theta = \{v, u\}$
$R_\Theta = N_\Theta - \{v, u\}$
$M \leftarrow$ Map for node information
for all $i \in R_\Theta$ **do**
  $h_{v,i}, w_{avg}\langle v, i \rangle = f\langle v, i \rangle$
  $h_{u,i}, w_{avg}\langle u, i \rangle = f\langle u, i \rangle$
  $w^i_{avg} = \frac{1}{2}(w_{avg}\langle v, i \rangle + w_{avg}\langle u, i \rangle)$
  $h^i_{avg} = \frac{1}{2}(h_{v,i} + h_{u,i})$
  $M \leftarrow (encode(i, 1/w^i_{avg}, h^i_{avg}))$
end for
sort M
for i in M **do**
  $O_\Theta \leftarrow O_\Theta \cup i$
  if $|O_\Theta| = \Theta$ **then**
    break
  end if
end for
return $\langle O_\Theta \rangle$



(a) Extracted subgraph with edge weight and each node displays H_{avg} and W_{avg}. (b) Encoded scores using the values H_{avg} and $1/W_{avg}$ (c) Sorted labels in a consistence way.

Figure 4. The representation of subgraph labeling based on encoding method. The left most figure represents the extracted subgraph with edge weight. Nodes with different colors indicates the various average hop distance (equal H_{avg} has same color), followed by average weight. The middle figure shows the encoded values and the right most figure represents the final labeling.

We now order remaining nodes list based on encoded value and store them until the total nodes equal to threshold value Θ. Algorithm 2 shows the step by step labeling process for labeling. For example, Figure 4 represents the process of subgraph node labeling. The leftmost figure illustrates a subgraph from the last snapshot of a given dynamic network for the target link $\langle 12 \rangle$. The nodes display the information of average hop (H_{Avg}) and average weight (W_{Avg}). We encoded these values to generate a unique code as shown in second rightmost Figure 2. Finally, every node gets a unique label after ordering encoding values.

C. Feature Matrix Construction

We have a series of snapshots of a given dynamic network. The subgraph extraction and node labeling have been processed at the last snapshot t. The same subgraph should have evolved throughout the time series $t_k, t_{k-1}, \ldots, t_t$. We therefore construct feature metrics for a subgraph of each snapshot. As we already discussed in the previous section, we build feature matrices of CN, JC, AA, PA, RA and Weight.

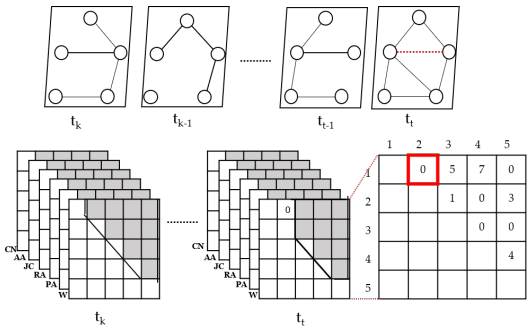


Figure 5. Example: (top) the representation of how a subgraph evolving through each snapshot, and (bottom) the way how adjacency matrices are created and the enlarged form of adjacency matrix for weight graph.

In figure 5, top figures illustrates the way how a subgraph evolves through time. We keep the same vertices of the subgraph in every snapshot and examine the evolution of links. We need to construct 6 feature matrices for the subgraph in each snapshot. Totally we construct $6 \times k$ number of feature matrices for a subgraph, where k is the time steps considered in our case.

Algorithm 3 describes the process of feature matrices construction. We create empty adjacency matrix lists to store the features of CN, JC, AA, PA, RA, W in each snapshot as below;

$$\{A^k_{l\times l}\langle cn\rangle, A^k_{l\times l}\langle jc\rangle, A^k_{l\times l}\langle aa\rangle, A^k_{l\times l}\langle pa\rangle, A^k_{l\times l}\langle ra\rangle, A^k_{l\times l}\langle w\rangle\}$$
$$\{A^{k-1}_{l\times l}\langle cn\rangle, A^{k-1}_{l\times l}\langle jc\rangle, A^{k-1}_{l\times l}\langle aa\rangle, A^{k-1}_{l\times l}\langle pa\rangle, \ldots, A^{k-1}_{l\times l}\langle w\rangle\}$$
$$\ldots$$
$$\{A^{t}_{l\times l}\langle cn\rangle, A^{t}_{l\times l}\langle jc\rangle, A^{t}_{l\times l}\langle aa\rangle, A^{t}_{l\times l}\langle pa\rangle, \ldots, A^{t}_{l\times l}\langle w\rangle\}$$

where l is the size of the ordered nodes list of the subgraph. The algorithm 3 continues until the above empty lists are filled by fetching the required information of node list from the lookup table.

In each last snapshot of the adjacency matrix of the weighted graph, we assign zero to the positive target link to hide the information of link existence. In Figure 5, the bottom figure illustrates how we construct the adjacency matrices. We fill only the upper triangle of the matrix to avoid duplicate values. At the last snapshot, we indicate with a red box where the value is always zero.

D. Modeling with CNN and LSTM

DLP-LES uses CNN and LSTM to model the link prediction framework. As described in the previous section, we have a sequence of adjacency matrices for a subgraph of a targeted link for the prediction from the last snapshot of a given dynamic network. The input data of DLP-LES is a sequence of adjacency matrices which is constructed as a form of $\Theta\times\Theta\times h$, where Θ is the number of nodes of the subgraph and h is the number of heuristic features used in our model. In DLP-LES, we treat each adjacency matrix as an image. We have the sequence of adjacency matrices in $t_k, t_{k-1}, \ldots, t_t$ as shown in Figure 5 bottom one. A sequence of images are really a video. So we can treat our model as a video classification problem,

Algorithm 3 Feature Matrix Construction

Input: Nodes ordered List O_Θ, Lookup Table $\langle\mathcal{R}\rangle$
Output: Feature Matrices $\langle\mathcal{F}\rangle = F_k\langle cn, jc, aa, pa, ra, w\rangle$, $F_{k-1}\langle cn, jc, aa, pa, ra, w\rangle, \ldots F_t\langle cn, jc, aa, pa, ra, w\rangle$

1: $l \leftarrow |O_\Theta|$
2: $A^k_{l\times l}[\,], A^{k-1}_{l\times l}[\,] \ldots A^t_{l\times l}[\,] = \{\}$
3: **for** $i \in O_\Theta$ **do**
4: **for** $j - i \in O_\Theta$ **do**
5: $\left.\begin{array}{l} A^k_{l\times l}[\,] = F_k\langle cn, jc, aa, pa, ra, w\rangle \\ A^{k-1}_{l\times l}[\,] = F_{k-1}\langle cn, jc, \ldots, w\rangle \\ \ldots \\ A^t_{l\times l}[\,] = F_t\langle cn, jc, aa, pa, ra, w\rangle \end{array}\right\} = F\langle i, j\rangle$
6: **end for**
7: **end for**
8: **return** $\langle\mathcal{F}\rangle$

where positive and negative links are two different classes. The positive links represent the link existence, $(v_i, v_j) \in \mathcal{E}_t$ while the negative links represent the absence of links between any two nodes, $(v_i, v_j) \notin \mathcal{E}_t$. To train the classifier, we build a dataset using last snapshot of the dynamic network with all existing links for the positive link class and the same number of non-existing links by using downsampling technique.

CNN is well known for image classification. We leverage this character to learn and extract features from each image, in our case each adjacency matrix. We first feed the input data to convolutional layers to extract the features and then pass those sequences to a separate LSTM to learn the long-range temporal dependencies from input sequences. In the CNN model, we use Rectified Linear Units (ReLu) as the activation function, which is computed using $f(x) = max(0, x)$, where, x is the input data. In the LSTM model and the output layer, we use sigmoid activation function, $\sigma(x) = \frac{1}{1+e^{(-x)}}$. In DLP-LES, we assign the binary cross-entropy for loss function to measure the performance of a classification model. It can be written as $-(y.log(p) + (1-y).log(p))$, where y is the label, p is predicted probability.

VI. EXPERIMENTAL RESULTS

To evaluate the effectiveness of our model, we perform experiments with five real-world dynamic social networks.

A. Datasets

We use five benchmark real world dynamic networks to test our model: **Enron** corpus [29] is an email communication network from the senior management of Enron for 6 months with 151 nodes and 50571 edges. Each node represents an employee and link represents email sent among employees. **Radoslaw** [30] is also an email communication network of a mid-sized manufacturing company from 2010-01-01 to 2010-09-30 with 167 nodes and 82900 edges. **Contact** [31] represents data from wireless devices carried by people with 274 nodes and 28200 edges. Every node specifies people, and a link appeared when they contacted with a timestamp which recorded every 20 seconds for 4 days. **College Messages** [32]

Table I COMPARISON OF AUC WITH STANDARD BASELINE METHODS FOR DYNAMIC NETWORK LINK PREDICTION.

Dataset	CN	JC	AA	PA	node2vec	LINE	DeepWalk	SDNE	DLP-LES
Enron	0.8106	0.8751	0.8970	0.8442	0.7596	0.5042	0.7190	0.9437	**0.9769**
Radoslaw	0.8417	0.8307	0.9028	0.8753	0.7417	0.6153	0.7342	0.8709	**0.9330**
Contact	0.8457	0.9141	0.9142	0.9027	0.8741	0.7360	0.8451	0.9376	**0.9913**
CollegeMessages	0.5742	0.5774	0.5843	0.5901	0.7049	0.4905	0.7506	0.7806	**0.9852**
EU-core	0.9227	0.9302	0.9341	0.7553	0.8602	0.6587	0.8201	0.9574	**0.9729**

contain private messages sent on an online social network at the University of California, Irvine with 1899 nodes and 59835 edges. The edge has the timestamp t, the time any two people contacted each others. **EU-Core** [33] is an email information from a large European research institution with 986 nodes and 332334 links. The node represents the members from 4 different departments and the links are the communications among them.

B. Performance Evaluation

We evaluate the effectiveness of DLP-LES model by comparing it with simple heuristic methods: CN, JC, AA, PA and network embedding methods: DeepWalk, node2vec, LINE and SDNE.

1) DeepWalk [13]: Random walks is used to learn latent representations, and considers vertices from second order proximity.
2) node2vec [12]: It learned by mapping of nodes to a low-dimensional space of features to maximizes the probability of preserving network proximity of nodes.
3) LINE [14]: It is suitable for any type of networks, including large scale networks. It used edge-sampling method to learn both the local and global network structures.
4) SDNE [34]: It is a semi-supervised deep model, and used both the first-order and second-order proximities together in an autoencoder based deep model.

For the implementation of the network embedding methods, we use the original source code by the author for node2vec[1], LINE[2], DeepWalk[3] and SDNE[4].

Evaluation Metric: Area Under the Curve (AUC) [35] is the standard evaluation metric in both static and dynamic link prediction problem. AUC estimates the probability that the predictor gives a higher score to a randomly chosen positive link than a randomly chosen negative link. The larger the AUC is, the better the model performs. The AUC can be defined as,

$$AUC = \frac{n' + 0.5n"}{n} \tag{12}$$

where n specifies the number of Independence comparisons, n' specifies the number of times that the positive link gets a higher probability score than the negative link, and n" specifies the number of times when they are equal.

[1]https://github.com/aditya-grover/node2vec
[2]https://github.com/tangjianpku/LINE
[3]https://github.com/phanein/deepwalk
[4]https://github.com/xiaohan2012/sdne-keras

C. Experimental Procedure

We use Python3 to implement the DLP-LES model. The data processing is conducted on IBM cluster with the specification of POWER8 52 processor 256 GB of RAM. We trained and tested our model in an Nvidia GTX 1050Ti, 4 GB GPU with 768 CUDA cores.

In DLP-LES, the CNN model contains 32 filters of size 5×5 in the convolution layer. Afterwards, it is sent to the average pooling with the size 3×3. The output is flattened before feeding into the LSTM model. The LSTM layer uses 128 internal cells. We train our neural network for 100 epochs with Adam optimizer algorithm. We assign 80% as training set, 10% as validation set, and 10% as testing set.

D. Results

The performance of the experimental setup for DLP-LES using CNN-LSTM is represented in Table I. The results are measured based on AUC in various benchmark dynamic datasets. DLP-LES outperforms all the standard state-of-the-art methods and most common heuristic methods. It also achieves above 97% of AUC in all tested dynamic networks except Radoslaw. However, DLP-LES reaches 93% of AUC in Radoslaw, which is higher than other compared methods. In College message, DLP-LES is remarkably outperformed than all baseline methods while others reach up to 78%.

The computational complexity of DLP-LES model can be expressed based on several factors. As we already explained in the above section, we first evaluate the heuristic values of each links in the given networks and stored as a lookup table. The complexity of subgraph extraction process is $O(kn)$, where k is the average hop number for the subgraph and n is the number of links. The complexity of feature matrix construction is $O(sn)$, because our feature matrix algorithm collects required information from the lookup table, which is $O(1)$, where $s = \Theta$ is the number of nodes in the subgraph.

VII. CONCLUSIONS

In this paper, we propose a novel framework DLP-LES, which is for link prediction problem in dynamic social networks based on effective subgraphs. This model uses the heuristic features of common neighbor based subgraph and

learns the evolving pattern throughout the considered time to predict future links. In this work, we introduce an encoding method for labeling subgraph consistently. To reduce the complexity, we construct a lookup table with all required information of links to use frequently through our proposed hash function. We also propose an algorithm for feature matrix construction, which is thereafter feed into CNN to extract the features and send to LSTM to learn long-term temporal feature of dynamic network. Since it analyzes the subgraph of a target link, this model has the advantage in applying for large-scale networks. We evaluate the performance of our model against the state-of-the-art methods and the basic heuristic method. DLP-LES achieves significantly high improvement than compared methods. Further, DLP-LES opens a new research direction in temporal network compression and expanding community detection in dynamic networks.

ACKNOWLEDGMENT

This research work was supported by International Business Machines (IBM); experiments were conducted on a high performance IBM Power System S822LC Linux Server.

REFERENCES

[1] T. Y. Berger-Wolf and J. Saia, "A framework for analysis of dynamic social networks," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 523–528.

[2] M. Kazemilari and M. A. Djauhari, "Correlation network analysis for multi-dimensional data in stocks market," *Physica A: Statistical Mechanics and its Applications*, vol. 429, pp. 62–75, 2015.

[3] L. Wang and J. Orchard, "Investigating the evolution of a neuroplasticity network for learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.

[4] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social networks*, vol. 25, no. 3, pp. 211–230, 2003.

[5] H. Sid Ahmed, B. Mohamed Faouzi, and J. Caelen, "Detection and classification of the behavior of people in an intelligent building by camera." *International Journal on Smart Sensing & Intelligent Systems*, vol. 6, no. 4, 2013.

[6] X. Wang, N. Gulbahce, and H. Yu, "Network-based methods for human disease gene prediction," *Briefings in functional genomics*, vol. 10, no. 5, pp. 280–293, 2011.

[7] M. E. Newman, "Clustering and preferential attachment in growing networks," *Physical review E*, vol. 64, no. 2, p. 025102, 2001.

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

[9] X. Li, N. Du, H. Li, K. Li, J. Gao, and A. Zhang, "A deep learning approach to link prediction in dynamic networks," in *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 2014, pp. 289–297.

[10] M. Rahman and M. Al Hasan, "Link prediction in dynamic networks using graphlet," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2016, pp. 394–409.

[11] J. Chen, J. Zhang, X. Xu, C. Fu, D. Zhang, Q. Zhang, and Q. Xuan, "E-lstm-d: A deep learning framework for dynamic network link prediction," *arXiv preprint arXiv:1902.08329*, 2019.

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

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

[14] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[15] M. Kaya, M. Jawed, E. Bütün, and R. Alhajj, "Unsupervised link prediction based on time frames in weighted–directed citation networks," in *Trends in Social Network Analysis*. Springer, 2017, pp. 189–205.

[16] C. Chiu and J. Zhan, "Deep learning for link prediction in dynamic networks using weak estimators," *IEEE Access*, vol. 6, pp. 35 937–35 945, 2018.

[17] M. Zhang and Y. Chen, "Weisfeiler-lehman neural machine for link prediction," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 575–583.

[18] Zhang, Muhan, and Y. Chen, "Link prediction based on graph neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 5165–5175.

[19] K.Ragunathan, K.Selvarajah, and Z.Kobti, "Link prediction by analyzing common neighbors based subgraphs using convolutional neural network." in *ECAI*, 2020.

[20] Yao, Lin, L. Wang, L. Pan, and K. Yao, "Link prediction based on common-neighbors for dynamic social network," *Procedia Computer Science*, vol. 83, pp. 82–89, 2016.

[21] S. Gao, L. Denoyer, and P. Gallinari, "Temporal link prediction by integrating content and structure information," in *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 2011, pp. 1169–1174.

[22] W. Yu, W. Cheng, C. C. Aggarwal, H. Chen, and W. Wang, "Link prediction with spatial and temporal consistency in dynamic networks." in *IJCAI*, 2017, pp. 3343–3349.

[23] X. Ma, P. Sun, and Y. Wang, "Graph regularized nonnegative matrix factorization for temporal link prediction in dynamic networks," *Physica A: Statistical mechanics and its applications*, vol. 496, pp. 121–136, 2018.

[24] T. Li, J. Zhang, S. Y. Philip, Y. Zhang, and Y. Yan, "Deep dynamic network embedding for link prediction," *IEEE Access*, vol. 6, pp. 29 219–29 230, 2018.

[25] K. Lei, M. Qin, B. Bai, G. Zhang, and M. Yang, "Gcn-gan: A non-linear temporal link prediction model for weighted dynamic networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 388–396.

[26] C. A. Bliss, M. R. Frank, C. M. Danforth, and P. S. Dodds, "An evolutionary algorithm approach to link prediction in dynamic social networks," *Journal of Computational Science*, vol. 5, no. 5, pp. 750–764, 2014.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] J. Chen, X. Xu, Y. Wu, and H. Zheng, "Gc-lstm: Graph convolution embedded lstm for dynamic link prediction," *arXiv preprint arXiv:1812.04206*, 2018.

[29] B. Klimt and Y. Yang, "The enron corpus: A new dataset for email classification research," in *European Conference on Machine Learning*. Springer, 2004, pp. 217–226.

[30] R. Rossi and N. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

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

[32] P. Panzarasa, T. Opsahl, and K. M. Carley, "Patterns and dynamics of users' behavior and interaction: Network analysis of an online community," *Journal of the American Society for Information Science and Technology*, vol. 60, no. 5, pp. 911–932, 2009.

[33] A. Paranjape, A. R. Benson, and J. Leskovec, "Motifs in temporal networks," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 601–610.

[34] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 1225–1234.

[35] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve." *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.