24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Link Prediction by Analyzing Common Neighbors Based Subgraphs using Convolutional Neural Network

Kumaran Ragunathan, Kalyani Selvarajah and Ziad Kobti[1]

Abstract. Link prediction (LP) in social networks is to infer if a link is likely to be formed in the future. Social networks (SN) are ubiquitous and have different types such as human interaction and protein-protein networks. LP uses heuristic methods including common neighbors and resource allocation to find the formation of future links. These heuristics are sensitive to different types of social networks. Certain types of heuristics work better for some SN types, but not for others. Selecting the appropriate heuristic method for the SN type is often a trial and error process. Recent ground-breaking methods, WLMN and SEAL, demonstrated that this selection process can be automated for the different types of SN. While these methods are promising, in some types of SN they still suffer from low accuracy. The objective of this paper is to address this weakness by introducing a novel framework called PLACN that incorporates the analysis of common neighbors of nodes on target link and a combination of heuristic features through a deep learning method. PLACN is driven by a new method to efficiently extract the subgraphs for a target link based on the common neighbors. Another novelty is the method for labeling subgraphs based on the average hop and average weight. Furthermore, we introduce a method to evaluate the approximate number of nodes in the subgraph. Our model converts link prediction to an image classification problem and uses a convolutional neural network. We tested our model on seven real-world networks and compared against traditional LP methods as well as two recent state-of-the-art methods based on subgraphs. Our results outperformed those LP methods reaching above 96% of AUC in benchmark SNs.

1 INTRODUCTION

Link prediction (LP) is of great interest due to its practical value in real-world applications such as e-commerce and friends recommendations, as well as networks for discovering future collaborators. The LP problem aims to predict the links that are expected to occur or not occur in the future. While LP has been discussed over the last two decades, the work of Jon Kleinberg and David Liben-Nowell has had a significant impact on this topic [23], drawing a great deal of attention in recent years. The traditional approaches include heuristic methods such as Common neighbors (CN) [28], Adamic Adar (AA)[2], and Resource Allocation (RA)[41]. Other approaches include supervised learning methods, such as SVM, bagging, and Naive Bayes [5].

Although many sophisticated methods for LP have been proposed, we have identified that for certain types of networks there exists simple heuristic methods, or combination of such heuristics, that can produce more accurate results. For any given heuristic it will not perform with the same accuracy in every different type of network. The reason is that these methods work based on the extracted pattern from the network topology, which may vary from one SN to another. This is a significant drawback with heuristic methods. Selecting the appropriate heuristic method for the SN type is often a trial and error process. Weisfeiler-Lehman Neural Machine (WLNM) method [39] proposed a solution to find the appropriate methods automatically, based on the extracted subgraph in its neighborhood. WLNM is considered to be a state-of-the-art link prediction method based on its high accuracy.

The WLNM used high-order heuristics, such as Katz index [17] and Pagerank [23], to achieve significant accuracy. However, this requires a large number of hops that span the enclosing subgraph to the entire network and requires additional computation time and memory. To overcome this issue, SEAL (learning from Subgraphs, Embeddings, and Attributes for Link prediction) proposed a method to learn general graph structure features from local enclosing subgraphs using Graph Neural Network (GNN)[40]. SEAL derived γ-$decaying\ theory$ to prove that a small number of hops is enough to extract the high-order heuristics and to achieve better accuracy than WLNM. However, we discover that SEAL also has many shortcomings.

Our both theoretical and empirical results motivate us to model a new link prediction framework that fixes the various shortcomings of SEAL. First, SEAL considered the h-hop enclosing subgraph for a pair of nodes (i, j) from the network $G = (V, E)$ by the set of nodes from i and j's neighbors up to h-hop, which may or may not include essential nodes for predicting the target link between nodes i and j. In our proposed model, PLACN described below, we construct subgraphs based on common neighbors of nodes i and j in different orders, which belong to the target link. The subgraphs include a high number of essential nodes, as common neighbors are shared by both nodes i and j. SEAL then evaluated a vector X for i and j with various heuristic features and an adjacency matrix of the enclosing subgraph A to feed to GNN. SEAL considered various features for only the nodes i and j, which belong to the target link. They ignore the heuristic features of all other nodes from the subgraph. Considering the heuristic features of the entire subgraph has a large impact on introducing a new link between any two nodes. As our contribution, we consider the heuristic features of the entire subgraph to incorporate the impact of every node from the subgraph.

Moreover, labeling each subgraph is a necessary task to handle the ordering of graph nodes which keeps consistency of all subgraphs. SEAL did not handle subgraph labeling well when common neighbors were in the subgraph. We discuss this further in subgraph labeling section. In PLACN, we introduce a new labeling method based

[1] School of Computer Science, University of Windsor, ON, Canada. email: ragunat@uwindsor.ca, selva111@uwindsor.ca and kobti@uwindsor.ca

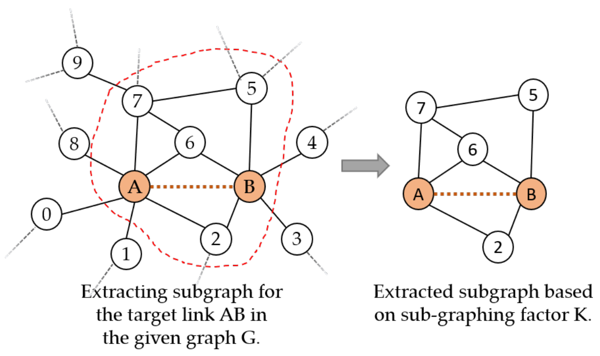

9
7
5
8
6
4
A
B
0
1
2
3
Extracting subgraph for the target link AB in the given graph G.
7
5
6
A
B
2
Extracted subgraph based on sub-graphing factor K.

Figure 1. The representation of subgraph extraction from the given graph G(the left figure), then the resultant extracted subgraph

on both the number of hops and weight of links to overcome the labeling inconsistency problem.

As our framework gives special importance to the heuristic features of common neighbors of target nodes to predict the links, we named our model PLACN (Predicting Links by Analyzing Common Neighbors).

We summarize our contributions as follows:

1. We propose a new method for constructing subgraphs;
2. We introduce a new theory to find the number of nodes K to include in a subgraph;
3. We propose a new labeling algorithm to maintain the consistent sequence of the adjacency matrix;
4. We extract heuristic features of nodes for the entire subgraph and model a new link prediction framework based on Convolutional Neural Network (ConvNet);
5. We demonstrate that PLACN outperforms all the heuristic methods as well as the state-of-the-art methods WLNM and SEAL.

2 Related Work

Link Prediction (LP) is a generic task for analyzing network data which appears in many applications including Recommending friends [7, 38], Co-author recommendation [6, 10, 15] and web-site link [16, 26, 27]. In SNs, it has got a significant attention after the innovative work by LibenNowell and Kleinberg (2007). Many of these applications considered the similarity measures. Generally, most of the approaches to LP problem can be categorized into supervised[5] and unsupervised learning framework[23]. The unsupervised learning includes topological feature-based methods such as CN[28], JCD[23], Katz index[2] and Pagerank[23]. The supervise learning methods includes SVM [5], decision tree [35] and random walk [7], focus more on individual nodes than network topologies.

There are a few approaches in the literature in terms of subgraph feature methods. The authors in [12] built several feature subsets according to their characteristics and evaluated classifier performance, and used several machine-learning algorithms for each one of these subsets for LP problem. The authors in [24] proposed a method based on a restrictive representation of the local topological embedding of two nodes, describing the relationship of them in terms of their common membership in all possible subgraphs. However, recently, WLNM [39] was proposed, initiated an idea to use deep learning method, and learned patterns through subgraphs to predict links. It outperformed than heuristic methods and other famous approaches, and considered as a state-of-the-art method. However, later, Zhang et al. [40] proposed a model SEAL closest to WLMN, and stated that WLMN has several drawbacks and need to resolve. SEAL proposed a new approach for subgraph extraction based on the hop numbers and subgraph labeling. They claimed that their method solved the drawbacks in WLNM model, and achieved better accuracy than WLNM. However, SEAL also has several drawbacks in terms of subgraph extraction and labeling. Both SEAL and WLMN didn't analyze the importance of common neighbors properly, which highly influence for the link occurrence. Therefore, we proposed PLACN model to overcome those drawbacks, and propose a new approach to analyze common neighbors.

3 PRELIMINARIES

3.1 Problem Definition

We are given an undirected, weighted graph $G(V, E)$ at a particular time t, representing the topological structure of a social network in which V is the set of vertices, E is the set of links, and the weight is based on interaction between any two nodes u and v. $A = (a_{i,j})_{N\times N}$ represents an adjacency matrix of the graph G, where $a_{i,j} > 0$ if i and j interact each others and $a_{i,j} = 0$ otherwise. We aim to build a model using ConvNet based on various heuristic features to predict whether the connection between u and v has a high probability of existing or not in the near future at time $t'(> t)$.

3.2 Heuristic Methods for Link Prediction

Generally, heuristic link prediction methods are based on topological structures which can be neighbor-based, path-based or random-walk based [36]. The neighbor-based include Common Neighbors (CN) [28], Jaccard Coefficient (JC), Adamic-Adar Coefficient (AA) [2], Preferential Attachment (PA) [8], and Resource Allocation (RA) [41] while other topological heuristic methods, sometimes called as higher-order heuristic, include Katz index [17], PageRank [23] and PropFlow [25].

In our model, we consider only the following neighbor-based (low-order) heuristic methods:

- Common Neighbors $|\Gamma(i) \cap \Gamma(j)|$
- Jaccard Coefficient $\frac{|\Gamma(i)\cap\Gamma(j)|}{|\Gamma(i)\cup\Gamma(j)|}$
- Adamic-Adar $\sum_{k\in|\Gamma(i)\cup\Gamma(j)|} \frac{1}{log|\Gamma(k)|}$
- Preferential Attachment $|\Gamma(i).\Gamma(j)|$
- Resource Allocation $\sum_{k\in|\Gamma(i)\cup\Gamma(j)|} \frac{1}{|\Gamma(k)|}$

where $\Gamma(i)$ and $\Gamma(j)$ represent the neighbor set of vertices for i and j respectively.

3.3 Convolutional Neural Network

Convolutional neural network (ConvNet) is comprised of one or more convolutional layers, and then followed by one or more fully connected layers. ConvNet is famous for image recognition [20, 14, 22] and image classifications [18, 21, 30]. We use ConvNet in our framework to predict the links in the future. Because the architecture

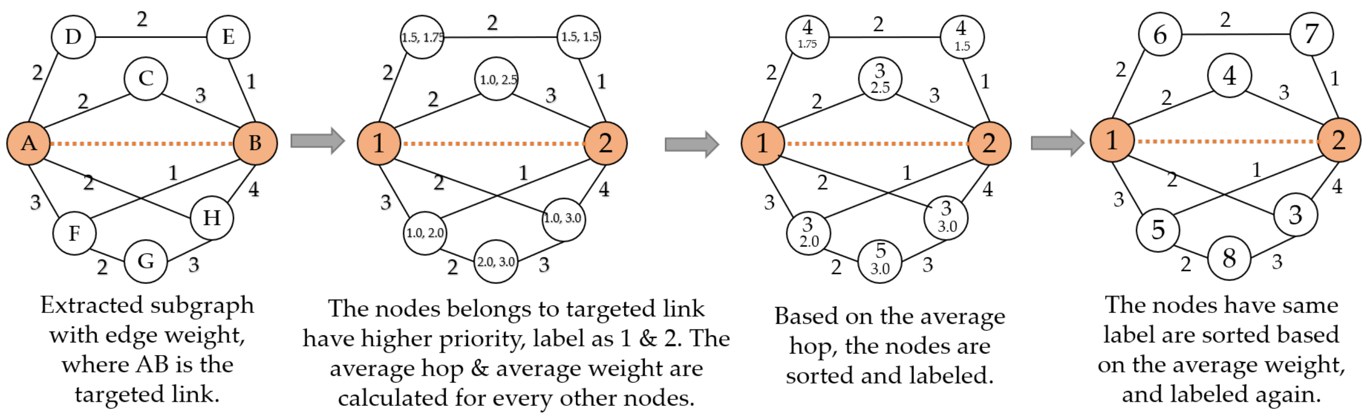

Extracted subgraph with edge weight, where AB is the targeted link.
The nodes belongs to targeted link have higher priority, label as 1 & 2. The average hop & average weight are calculated for every other nodes.
Based on the average hop, the nodes are sorted and labeled.
The nodes have same label are sorted based on the average weight, and labeled again.

Figure 2. The representation of labeling: subgraph extraction (the leftmost figure) for the target link AB, step by step process when labeling, and final labeled subgraph (the rightmost figure).

Algorithm 1 Subgraph Extraction

Input: Target link E_{ij}, Graph $G(E, V)$, and Expected number of nodes in subgraph K.
Output: Subgraph $\langle S \rangle$ for the link E_{ij}.

```
1: $N_K = \{i, j\}$
2: $N_{temp} = \{\}$
3: $h = 1 \leftarrow$ number of order
4: while $|N_K| < K$ do
5:     $N_{temp} = \Gamma^h(i) \cap \Gamma^h(j)$
6:     $N_K = N_K \cup N_{temp}$
7:     $h \leftarrow h + 1$
8: end while
9: $\langle S \rangle = subgraphG(N_K)$
10: return $\langle S \rangle$
```

of the ConvNet is designed to take benefit of 2D structure of input images, the heuristic features of the subgraph's adjacency matrices can be treated as the 2D structure in our model. The input of our model would be $K \times K \times r$, where K is the number of nodes of the subgraph and r represents the number of heuristic features considered in our model.

4 PLACN MODEL

In this section, we describe our novel PLACN framework. Since ConvNet performed well in the image classification we utilize this characteristic in our framework. So, we convert the link prediction as an image classification problem, where positive and negative links are two different classes. PLACN includes the following main steps:

1. Evaluating the Sub-graphing Factor K in the subgraph.
2. Extracting the subgraph $\langle S \rangle$ for a target link.
3. Labeling the extracted subgraph.
4. Constructing feature matrices.
5. Training convolutional neural networks.

4.1 Number of Nodes in a Subgraph

Generally, SN has various topological structures. In some networks vertices are densely connected while, in other networks, they are sparsely connected. Based on this, the average node degree of an undirected graph $\frac{2|E|}{|V|}$ changes over different networks. PLACN focuses strongly on subgraphs of common neighbors of the target link. There is a need for deciding an optimal number of nodes K in the subgraphs rather than defining it by the user. There are two primary purpose of defining optimal number of nodes. First, it reduces the computational cost because subgraphs will be extracted with the optimal number of nodes. Second, the number of nodes can be related to the network properties. In a given SN G, the number of neighbors is proportional to the average node degree. The node degree is only based on first-hop neighbors. We need the information of high order common neighbors (CN). Because we can't consider the full set of high-order CN, we examine a fraction of it using K value. Therefore, we defined K with both concepts, average node degree and its fraction, as below.

$$\begin{aligned} K &\approx \text{AvgNodeDegree} + \text{AvgNodeDegree} \times \text{NetworkDensity} \\ &\approx \text{AvgNodeDegree} \ \ (1 + \text{NetworkDensity}) \end{aligned}$$

The above relationship can be mathematically formulated as;

$$K \approx \left\lceil \frac{2|E|}{|V|} \left(1 + \frac{2|E|}{|V|(|V|-1)} \right) \right\rceil \tag{1}$$

where $\frac{2|E|}{|V|}$ is the average node degree, and $\frac{2|E|}{|V|(|V|-1)}$ is the density of the given network. Since the calculated value might be a floating number, we take the ceiling of the value.

We test our formula on various datasets and empirically verify that the calculated K is an optimal value for achieving better AUC (Area Under Curve) in most cases with the minimum computational cost. We test our model by varying different K values, and then plot a graph for AUC. Figure 6 shows the empirical results when testing various K values on different networks. We will discuss further details in the experimental results section.

Algorithm 2 Subgraph Labeling

```
Input: Nodes List $N_K$, Target link $E_{ij}$, Subgraph $\langle S\rangle$
Output: Ordered nodes list $O_K$

 1: $O_K = \{i, j\}$
 2: $R_K = N_K - \{i, j\}$
 3: $M\langle k : \langle w_{avg}, h_{avg}\rangle\rangle \leftarrow$ Map for node information
 4: for all $v \in R_K$ do
 5:     $h_i = min(dist(v, i))$
 6:     $h_j = min(dist(v, j))$
 7:     $w^v_{avg} = \frac{1}{2}\left(\frac{1}{h_i}\sum_{p=0}^{h_i} w^i_p + \frac{1}{h_j}\sum_{p=0}^{h_j} w^j_p\right)$
 8:     $h^v_{avg} = \frac{1}{2}(h_i + h_j)$
 9:     $M \leftarrow (v : (1/w^v_{avg}, h^v_{avg}))$
10: end for
11: sort the map based on $h_{avg}$
12:         sort the map based on $w_{avg}$ for same $h_{avg}$
13: for $v$ in $M$ do
14:     $O_K \leftarrow O_K \cup v$
15: end for
16: if $|O_K| > K$ then
17:     while $|O_K| = K$ do
18:         remove nodes from bottom
19:     end while
20: end if
21: return $\langle O_K\rangle$
```

4.2 Subgraph Extraction

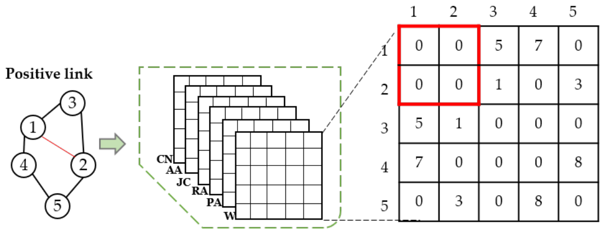
Subgraph extraction is a process of collecting nodes, which relate to the target link. SEAL constructed the subgraph based on the *h-hop* neighbors for the target nodes i and j. The entire nodes of the subgraph in SEAL might not have a great influence on both nodes i and j. At the same time, only a few of them might be common for both the nodes i and j. This is a major drawback of SEAL. As SEAL highly relies on the number of hops h, when the hop number increases, the number of nodes in the subgraph subsequently increases. This leads to memory and computation cost issues.

Instead of constructing the subgraphs based on the hops, PLACN introduces an approach to construct the subgraph only with the common neighbors of both nodes i and j. It drives the subgraphs with the set of nodes which interact with both nodes i and j. The number of nodes K in the subgraph $\langle S\rangle$ can be approximately estimated, which is explained in the previous section. The process of subgraph extraction of $\langle S\rangle$ is described in algorithm 1.

In algorithm 1, for a given link between i and j from graph G, we first collect their first order common neighbors $\Gamma^1(i) \cap \Gamma^1(j)$ to a node list N_K. Then, we add vertices in $(\Gamma^2(i) \cap \Gamma^2(j)), (\Gamma^3(i) \cap \Gamma^3(j))$, ..., iteratively until $|N_K| \geq K$, where $\Gamma^p(q)$ is the pth order neighbor nodes of node q.

At the end of algorithm 1, the number of nodes in N_K might be equal to K, or $|N_K| > K$. However, maintaining the consistency of the size of each subgraph is crucial to train the ConvNets. Therefore, we remove some nodes from $|N_K|$ until $|N_K| = K$. For this procedure, we adopt a different strategy which we explain in the subgraph labeling section. As a result, the size of each subgraph will be K.

For example, as shown in figure 1, we are trying to extract the subgraph for target link AB when the size of the subgraph $K = 6$. First order common neighbors are $\{2, 6\}$, and second-order common neighbors are $\{5, 7\}$. So, the extracted subgraph for the target link AB has the list of vertices $\{A, B, 2, 5, 6, 7\}$.



$$\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 0 & 0 & 5 & 7 & 0 \\
2 & 0 & 0 & 1 & 0 & 3 \\
3 & 5 & 1 & 0 & 0 & 0 \\
4 & 7 & 0 & 0 & 0 & 8 \\
5 & 0 & 3 & 0 & 8 & 0
\end{array}$$

Figure 3. The representation of constructing feature matrices: labeled subgraph (the leftmost figure) for a positive target link, and enlarged representation of an adjacency matrix of weight (the rightmost figure)

Note that if two nodes are from disconnected components of the network, we will have both the source and destination node in the subgraph. So, the subgraph will never be empty. We will calculate the heuristics for source and destination nodes. Other information for $(k - 2)$ nodes will be zero in the feature matrices.

4.3 Subgraph Labeling

Subgraph labeling is a process of ordering the nodes in a sequence. This process is essential to maintain the information uniformly among all subgraphs because machine learning models read data sequentially. The method used in SEAL, Double-Radius Node Labeling, labeled the subgraph nodes based on the number of hops between nodes i and j. This leads to a problem where the common neighbors in the same order will receive the same label. Because each subgraph that has the same common neighbor could be labeled in different ways, the adjacency matrices of the subgraphs become inconsistent. In PLACN, we propose a new strategy to process the subgraph labeling consistently as below.

Algorithm 2 describes the overall process for the subgraph labeling. After the extraction of the subgraph, nodes i and j belong to the target link. They will be the first two labeled positions of an ordered vertices list O_K. For each remaining node in R_K, where $R_K = N_K - \{i, j\}$, we need to calculate the average hop and average path weight with i and j. So, we first evaluate the minimum number of hops (i.e. the shortest path) h_i and h_j from i and j for R_K respectively. Then using equation 2, we calculate the average weight for each node in R_K.

$$w^v_{avg} = \frac{1}{2}\left(\frac{1}{h_i}\sum_{p=0}^{h_i} w^i_p + \frac{1}{h_j}\sum_{p=0}^{h_j} w^j_p\right) \tag{2}$$

where $\sum_{p=0}^{h_i} w^i_p$ evaluates the total weight of the shortest path between node $v \in R_K$ and i. Since each node in R_K has considerable influence on link ij, we combine the average weight of the shortest path from i and j. Similarly, we calculate the average hop for each node in R_k using the following equation 3.

$$h^v_{avg} = \frac{1}{2}(h_i + h_j) \tag{3}$$

At this point, every node in R_K has both the average weight (w_{avg}) and average hop (h_{avg}) values. We first order them based on the h_{avg} values. If any two nodes x, $y \in R_K$ have the same average hop, that is $h^x_{avg} = h^y_{avg}$, we then order them based on w_{avg} in descending order and store them in the ordered list O_K. As we discussed in the subgraph labeling, the adjacency matrix of each

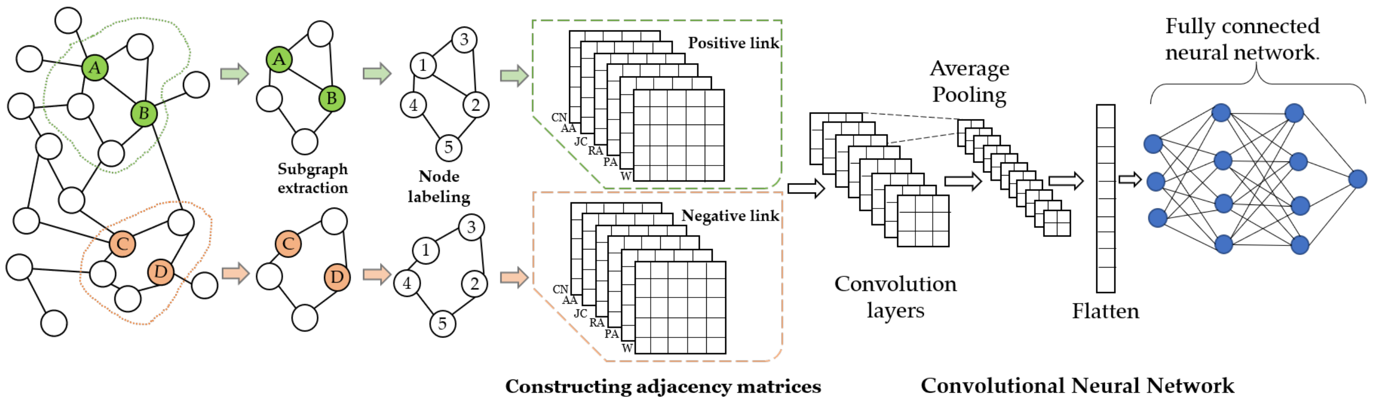


Figure 4. The architecture of PLACN model.

subgraph must maintain the same size K. Therefore, if $|O_K| > k$, we discard the nodes from the bottom until $|O_K| = k$.

For example, figure 2 illustrates the labeling process for the extracted subgraph for the target link AB. The leftmost figure shows the extracted graph for the target link AB. So, we label the nodes in target link AB as 1 and 2 in the next diagram. The remaining nodes show the calculated average hop and average weight. The third diagram shows the labeling based on average hop, where $h^D_{avg} = h^E_{avg} = 1.5$ and $h^F_{avg} = h^H_{avg} = h^C_{avg} = 1.0$. Now, we check w_{avg} for $\{C, F, H\} = \{2.5, 2.0, 3.0\}$ and $\{D, E\} = \{1.75, 1.5\}$, then label them as shown in the rightmost figure. Eventually, every node in the subgraph will have a unique label.

4.4 Constructing Feature Matrices

As discussed in the preliminary section we consider five heuristic features including CN, JC, AA, PA, and RA. Each adjacency matrix of the subgraph can be represented as $A^{CN}_{(ij)}, A^{JC}_{(ij)}, A^{AA}_{(ij)}, A^{PA}_{(ij)}$ and $A^{RA}_{(ij)}$ ordered in sequence of the name for each of the heuristic methods. In addition to this, we include an adjacency matrix of the graph weight, where diagonal values are zeros. SEAL considers only this adjacency matrix without including weight and feature vector for the target link. PLACN constructs the heuristic feature matrices for each vertex in the subgraph. For this purpose, we use the information from the entire graph G. As illustrated in figure 3, the middle figure represents the sample adjacency matrices for the labeled subgraph. The red box in the rightmost figure indicates the target link and its value. We always assign zero to the positive target link in the adjacency matrix of the weighted graph. The reason is that when we test PLACN model, positive links should not contain any information of the link's existence. PLACN needs to learn both the positive and negative links without the links' existing information.

4.5 Training Convolutional Neural Network

In the above sections we discussed the subgraph extraction for the target link, subgraph labeling, and constructing adjacency matrices for heuristic features of the subgraph. Following this, PLACN incorporates the ConvNet as a classifier. In order to train the classifier, we construct a dataset with all existing links, where the node pair $i, j \in V$ such that $(i, j) \in E$, for the positive link class. To avoid the class imbalance problem, we used downsampling technique to create a negative link class. So, we randomly sampled the negative links, where random node pairs $i, j \in V$ such that $(i, j) \notin E$, as the same number of the positive links to create negative links class.

In the next step of PLACN, the adjacency matrices are normalized between 0 and 1 for each layer. Figure 5 exhibits the graphical representation of randomly selected normalized data from USAir Network. We then apply convolution operation, average pooling, and flatten the matrices to feed into a fully connected neural network. Figure 4 illustrates the overall architecture of PLACN model.

Since this is a binary classification problem, we set the binary cross-entropy loss function in ConvNet. The formula for the binary cross-entropy loss function can be written as:

$$\begin{aligned} BCE &= -\sum_{i=1}^{c'=2} t_i log(f(s_i)) \\ &= -t_i log(f(s_1)) - (1 - t_i) log(f(s_1)) \end{aligned}$$

where c' is the number of classes, s_1 and t_1 are score and the groundtruth label for the class c_1.

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	**Ave node Degree**
USAir	332	2126	12.81
NS	1589	2742	3.45
PB	1222	16714	27.36
Yeast	2375	11693	9.85
C.ele	297	2148	14.46
Power	4941	6594	2.67
Router	5022	6258	2.49

Table 1. The statistical information of each real-world networks.

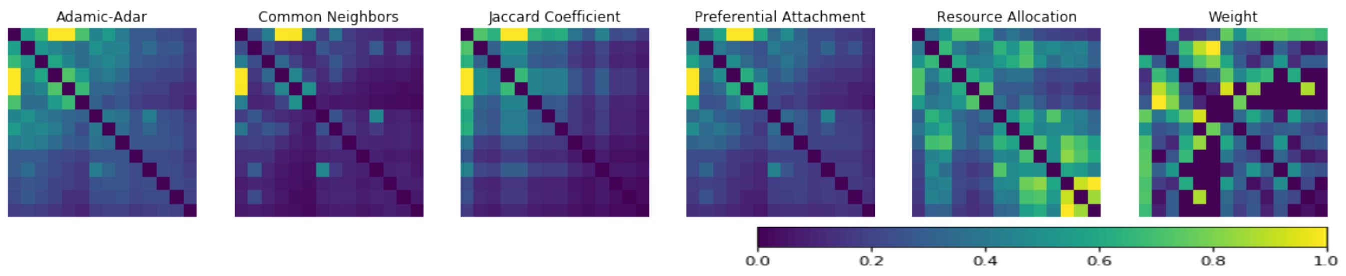


Figure 5. The representation of randomly selected data from USAir network.

5 EXPERIMENTAL RESULTS

We conduct the experiments with real-world networks to evaluate PLACN and use AUC (Area Under Curve) as evaluation metrics.

5.1 Datasets

We use seven real-world datasets in our experiments: **USAir** is a transportation network of US airlines [9], **NS** is a co-authorship information of researchers on network science [29], **PB** is a US political web-blogger connectivity data in 2005 [1], **Yeast** is a network of interactions between yeast proteins [34], **C.ele** is a neural network of neurons in C. elegans, a type of worm [37], **Power** is the power grid of the western US [37], and **Router** is a router level map of the Internet [32].

Table 1 comprises the statistical information of each network. We split each dataset into a training set (90%) and testing set (10%) from both positive and negative links classes.

5.2 Comparing with Other Methods

We first compare PLACN with seven popular heuristic methods based on topological structures:

- **Common Neighbors (CN)**: similarity-based method, which counts the number of neighbors that the two nodes have in common [28].
- **Jaccard Coefficient (JF)**: similar concept as CN, but it is a normalized form of common neighbours.
- **Preferential attachment (PA)**: measure to indicate that new links will be more likely to connect higher-degree nodes than lower ones [8].
- **Adamic-Adar (AA)**: similarity measure between two nodes by weighting fewer neighbors more heavily [2].
- **Resource allocation (RA)**: It is similar to AA. However, RA punishes the high-degree common neighbors more heavily than AA [41].
- **Katz index**: It considers all paths between two nodes and weighs shorter ones more heavily. [17].
- **PageRank (PR)**: It ranks a node in a graph which is proportional to the likelihood that the node will be reached through a random walk on the graph [23].

For Katz, we set the damping factor β to 0.001 [39]. For PageRank, we set the damping factor d to 0.85 [11]. Then, we compare with two state-of-the-art, which used subgraphs: WLNM and SEAL. For WLMN, we use the implementation at https://github.com/muhanzhang/LinkPrediction. For SEAL, we use the source code at https://github.com/muhanzhang/SEAL.

We then compare PLACN with four state-of-the-art latent feature methods:

- **Stochastic block model (SBM)**: an adaptation of mixture modeling to relational data [4].
- **node2vec (N2V)**: Word-to-vector approach with biased random walk [13].
- **LINE**: Network embedding method with objective function that preserves both the first-order and second-order proximities [33].
- **Variational graph auto-encoder (VGAE)**: a framework for unsupervised learning on graph-structured data based on the variational auto-encoder [19], is also uses subgraphs on their model.

For MF, we use the libFM software [31]. For SBM, we use the implementation of [3] at http://tuvalu.santafe.edu/ aaronc/wsbm/ using a latent group number 12. For N2V, we use the implementation at https://github.com/eliorc/node2vec. For LINE, we use the implementation at https://github.com/tangjianpku/LINE. For VGAE, we use the source code at https://github.com/tkipf/gae. For node2vec and LINE, we use the same setting as described in SEAL.

5.3 Experimental Setup

For the proposed PLACN model we test seven real-world networks with the calculated optimal K value of each network, as discussed in the previous section. In the ConvNet we use one convolution layer with 32 filters of size 4×4. We then apply the average pooling of size 2×2. In the following fully connected neural network two hidden layers with 128 neurons are used. We train our neural network for 50 epochs. During the training, we use 10% training set for the validation and saved the model, which has lowest validation lost as the best model. Finally, we use our saved model to predict the test data. We repeat the above process for 10 times and calculated the average AUC (μ_{AUC}) and standard deviation (σ_{AUC}).

Dataset	**CN**	**JC**	**PA**	**AA**	**RA**	**Katz**	**PR**	**WLNM**	**SEAL**	**PLACN**
USAir	94.04±1.02	89.82±1.49	89.76±1.23	94.91±1.28	95.79±0.98	92.92±1.34	93.96±1.17	95.95±1.10	96.33±0.82	**98.36±0.24**
NS	93.94±1.08	94.37±0.95	69.05±1.93	94.76±1.03	94.35±0.99	95.02±0.94	94.89±1.08	98.61±0.49	98.91±0.32	**99.53±0.13**
PB	92.08±0.31	87.39±0.46	90.28±0.37	92.30±0.39	92.51±0.26	92.89±0.45	93.60±0.38	93.49±0.47	94.70±0.34	**96.67±0.44**
Yeast	89.41±0.59	89.32±0.66	82.17±1.20	89.31±0.59	89.45±0.72	92.29±0.58	92.76±0.55	95.62±0.52	97.96±0.47	**98.87±0.30**
C.ele	84.97±1.72	80.19±1.47	74.83±1.94	87.01±1.38	87.49±1.41	86.42±1.71	90.32±1.52	86.18±1.72	90.15±0.86	**96.08±0.26**
Power	58.86±0.75	58.79±0.83	44.37±1.00	58.83±0.92	58.81±0.84	65.42±1.51	66.03±1.59	84.76±0.98	88.50±1.17	**98.78±0.38**
Router	56.58±0.41	56.39±0.46	47.62±1.51	56.39±0.51	56.39±0.51	38.62±1.42	38.82±1.31	94.41±0.88	96.45±0.96	**98.40±0.38**

Table 2. Comparison of AUC with heuristic methods, WLMN and SEAL.

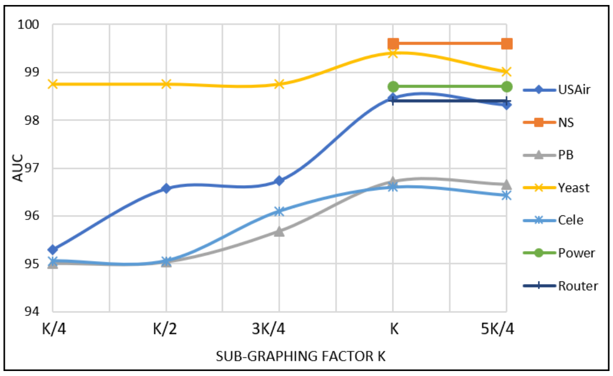


Figure 6. The representation of constructing feature matrices.

5.4 Results

Overall PLACN achieves a significant improvement compared to other methods in networks of different characteristics. In terms of AUC, PLACN performs remarkably better in Power and Router networks, in which the baseline and latent methods achieve below 80%, and recent state-of-the-art methods, WLNM and SEAL achieve up to 85% and 90% respectively. Moreover, PLACN has the AUC higher than 95% among all tested networks. Tables 2 and 3 represent the comparison of other methods with PLACN.

Tables 2 represents the AUC performance in the form of $\mu_{AUC} \pm \sigma_{AUC}$. For all seven networks we calculated the K value and tested the AUC by varying K for the extracting subgraphs as shown in fig 6. After we calculate K values, the AUC graph tends to converge for most cases. This pattern shows that the calculated K is the minimal optimal value for the different networks. Therefore, our formula for K is suitable in most cases. However, if the calculated $K < 5$, we assume $K = 5$, and process because it will consist of at least 3 common neighbors in subgraph.

When comparing the complexity of our algorithms, we evaluated based on 4 aspects: subgraph extraction, subgraph node labeling, feature extraction and classifier. In terms of subgraph extraction, the complexity of our algorithm ($O(n)$) is equal to SEAL. In graph labeling, SEAL uses a hash function, which led to duplicate labels for the same order common neighbors. Our algorithm uses sorting based on the average hop and weight, which has the complexity of Klog(K) (where K is sub-graphing factor). Even though our labeling method requires sorting, it tends to solve a key issue that is inconsistency in subgraphs. While SEAL includes the high-order heuristics, such as Katz, Pagerank and Simrank, which have the complexity $O(n^3)$, PLACN didn't include them. At the same time, PLACN is able to outperform only with the low-order heuristics. Finally, SEAL used DCGNN as a classifier that considered sort pooling as their aggregation layer, which required an additional time for computations. Our model reduces the cost while calculating the heuristics in the classification process, and trade-off the cost in labeling to keep consistency. PLACN therefore can be considered to have the same complexity as SEAL.

Dataset	**SBM**	**N2V**	**LINE**	**VGAE**	**PLACN**
USAir	94.85±1.14	91.44±1.78	81.47±10.71	89.28±1.99	**98.36±0.24**
NS	92.30±2.26	91.52±1.28	80.63±1.90	94.04±1.64	**99.53±0.13**
PB	93.90±0.42	85.79±0.78	76.95±2.76	90.70±0.53	**96.67±0.44**
Yeast	91.41±0.60	93.67±0.46	87.45±3.33	93.88±0.2	**98.87±0.30**
C.ele	86.48±2.60	84.11±1.27	69.21±3.14	81.80±2.18	**96.08±0.26**
Power	66.57±2.05	76.22±0.92	55.63±1.47	71.20±1.65	**98.78±0.38**
Router	85.65±1.93	65.46±0.86	67.15±2.10	61.51±1.22	**98.40±0.38**

Table 3. Comparison of AUC with Latent feature methods.

6 Conclusions

Link prediction is currently a popular research field and used in many application domains. Although heuristic methods are simple and show good results, intensive guess work is needed to manually match the right heuristic to the corresponding network type since the same heuristic will not always perform well in different networks. The state-of-the-art approaches failed to analyze the impact of common nodes on the target link. We implemented a novel framework PLACN that solved these issues by creating a new subgraph extracting algorithm and a new node labeling algorithm to learn from different networks automatically. We also derived an equation for sub-graphing factor K, which helps to determine subgraph size rather than relying on a user-defined value. These characteristics provide autonomy in our framework to learn and adopt the topological patterns of different networks. We tested our model with six state-of-the-art methods and baseline methods. The results showed that PLACN performs significantly better than the state-of-the-art methods as well as baseline methods and set new state-of-the-art performance. PLACN also enables new paths to further research on recommendation systems and knowledge graph completions.

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] Robert Ackland et al., 'Mapping the us political blogosphere: Are conservative bloggers more prominent?', in *BlogTalk Downunder 2005 Conference, Sydney*. BlogTalk Downunder 2005 Conference, Sydney, (2005).

[2] Lada A Adamic and Eytan Adar, 'Friends and neighbors on the web', *Social networks*, **25**(3), 211–230, (2003).

[3] Christopher Aicher, Abigail Z Jacobs, and Aaron Clauset, 'Learning latent block structure in weighted networks', *Journal of Complex Networks*, **3**(2), 221–248, (2014).

[4] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing, 'Mixed membership stochastic blockmodels', *Journal of machine learning research*, **9**(Sep), 1981–2014, (2008).

[5] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki, 'Link prediction using supervised learning', in *SDM06: workshop on link analysis, counter-terrorism and security*, (2006).

[6] Soner Aydın, *Link prediction models for recommendation in academic collaboration network of Turkey*, Ph.D. dissertation, 2017.

[7] Lars Backstrom and Jure Leskovec, 'Supervised random walks: predicting and recommending links in social networks', in *Proceedings of the fourth ACM international conference on Web search and data mining*, pp. 635–644. ACM, (2011).

[8] Albert-László Barabási and Réka Albert, 'Emergence of scaling in random networks', *science*, **286**(5439), 509–512, (1999).

[9] V Batagelj and A Mrvar, 'Pajek datasets http://vlado. fmf. uni-lj. si/pub/networks/data/mix', *USAir97. net*, (2006).

[10] Michele A Brandão, Mirella M Moro, Giseli Rabello Lopes, and José PM Oliveira, 'Using link semantics to recommend collaborations in academic social networks', in *Proceedings of the 22nd International Conference on World Wide Web*, pp. 833–840. ACM, (2013).

[11] Sergey Brin and Lawrence Page, 'The anatomy of a large-scale hypertextual web search engine', *Computer networks and ISDN systems*, **30**(1-7), 107–117, (1998).

[12] Michael Fire, Lena Tenenboim, Ofrit Lesser, Rami Puzis, Lior Rokach, and Yuval Elovici, 'Link prediction in social networks using computationally efficient topological features', in *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pp. 73–80. IEEE, (2011).

[13] Aditya Grover and Jure Leskovec, 'node2vec: Scalable feature learning for networks', in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864. ACM, (2016).

[14] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen, 'Convolutional neural network architectures for matching natural language sentences', in *Advances in neural information processing systems*, pp. 2042–2050, (2014).

[15] Shiping Huang, Yong Tang, Feiyi Tang, and Jianguo Li, 'Link prediction based on time-varied weight in co-authorship network', in *Computer Supported Cooperative Work in Design (CSCWD), Proceedings of the 2014 IEEE 18th International Conference on*, pp. 706–709. IEEE, (2014).

[16] Shantha Jayalal, Chris Hawksley, and Pearl Brereton, 'Website link prediction using a markov chain model based on multiple time periods', *International Journal of Web Engineering and Technology*, **3**(3), 271–287, (2007).

[17] Leo Katz, 'A new status index derived from sociometric analysis', *Psychometrika*, **18**(1), 39–43, (1953).

[18] Yoon Kim, 'Convolutional neural networks for sentence classification', *arXiv preprint arXiv:1408.5882*, (2014).

[19] Thomas N Kipf and Max Welling, 'Variational graph auto-encoders', *arXiv preprint arXiv:1611.07308*, (2016).

[20] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back, 'Face recognition: A convolutional neural-network approach', *IEEE transactions on neural networks*, **8**(1), 98–113, (1997).

[21] Gil Levi and Tal Hassner, 'Age and gender classification using convolutional neural networks', in *Proceedings of the iEEE conference on computer vision and pattern recognition workshops*, pp. 34–42, (2015).

[22] Ming Liang and Xiaolin Hu, 'Recurrent convolutional neural network for object recognition', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3367–3375, (2015).

[23] David Liben-Nowell and Jon Kleinberg, 'The link-prediction problem for social networks', *Journal of the American society for information science and technology*, **58**(7), 1019–1031, (2007).

[24] Ryan N Lichtenwalter and Nitesh V Chawla, 'Vertex collocation profiles: subgraph counting for link analysis and prediction', in *Proceedings of the 21st international conference on World Wide Web*, pp. 1019–1028. ACM, (2012).

[25] Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla, 'New perspectives and methods in link prediction', in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 243–252. ACM, (2010).

[26] V Valli Mayil, 'Web navigation path pattern prediction using first order markov model and depth first evaluation', *International Journal of Computer Applications (0975-8887)*, **45**(16), (2012).

[27] Megha Mishra, Harsha Dubey, and Vishnu Kumar Mishra, 'Page navigation prediction based on longest common subsequence and apriori algorithm', *i-manager's Journal on Cloud Computing*, **3**(3), 27, (2016).

[28] Mark EJ Newman, 'Clustering and preferential attachment in growing networks', *Physical review E*, **64**(2), 025102, (2001).

[29] Mark EJ Newman, 'Finding community structure in networks using the eigenvectors of matrices', *Physical review E*, **74**(3), 036104, (2006).

[30] ME Paoletti, JM Haut, J Plaza, and A Plaza, 'A new deep convolutional neural network for fast hyperspectral image classification', *ISPRS journal of photogrammetry and remote sensing*, **145**, 120–147, (2018).

[31] Steffen Rendle, 'Factorization machines with libfm', *ACM Transactions on Intelligent Systems and Technology (TIST)*, **3**(3), 57, (2012).

[32] Neil Spring, Ratul Mahajan, and David Wetherall, 'Measuring isp topologies with rocketfuel', in *ACM SIGCOMM Computer Communication Review*, volume 32, pp. 133–145. ACM, (2002).

[33] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei, 'Line: Large-scale information network embedding', in *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077. International World Wide Web Conferences Steering Committee, (2015).

[34] Christian Von Mering, Roland Krause, Berend Snel, Michael Cornell, Stephen G Oliver, Stanley Fields, and Peer Bork, 'Comparative assessment of large-scale data sets of protein–protein interactions', *Nature*, **417**(6887), 399, (2002).

[35] Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Albert-Laszlo Barabasi, 'Human mobility, social ties, and link prediction', in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1100–1108. Acm, (2011).

[36] Peng Wang, BaoWen Xu, YuRong Wu, and XiaoYu Zhou, 'Link prediction in social networks: the state-of-the-art', *Science China Information Sciences*, **58**(1), 1–38, (2015).

[37] Duncan J Watts and Steven H Strogatz, 'Collective dynamics of 'small-world'networks', *nature*, **393**(6684), 440, (1998).

[38] Shuang-Hong Yang, Alexander J Smola, Bo Long, Hongyuan Zha, and Yi Chang, 'Friend or frenemy?: predicting signed ties in social networks', in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pp. 555–564. ACM, (2012).

[39] Muhan Zhang and Yixin Chen, 'Weisfeiler-lehman neural machine for link prediction', in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 575–583. ACM, (2017).

[40] Muhan Zhang and Yixin Chen, 'Link prediction based on graph neural networks', in *Advances in Neural Information Processing Systems*, pp. 5165–5175, (2018).

[41] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang, 'Predicting missing links via local information', *The European Physical Journal B*, **71**(4), 623–630, (2009).