Hyper-Substructure Enhanced Link Predictor

Jian Zhang
Institute of Cyberspace Security
Zhejiang University of Technology, China
jianzh@zjut.edu.cn

Jun Zheng
Institute of Cyberspace Security
Zhejiang University of Technology, China
zjun2878@gmail.com

Jinyin Chen
Institute of Cyberspace Security
Zhejiang University of Technology, China
chenjinyin@zjut.edu.cn

Qi Xuan
Institute of Cyberspace Security
Zhejiang University of Technology, China
xuanqi@zjut.edu.cn

ABSTRACT

Link prediction has long been the focus in the analysis of network-structured data. Though straightforward and efficient, heuristic approaches like *Common Neighbors* perform link prediction with pre-defined assumptions and only use superficial structural features. While it is widely acknowledged that a vertex could be characterized by a bunch of neighbor vertices, network embedding algorithms and newly emerged graph neural networks still exploit structural features on the whole network, which may inevitably bring in noises and limits the scalability of those methods. In this paper, we propose an end-to-end deep learning framework, namely *hyper-substructure enhanced link predictor* (HELP), for link prediction. HELP utilizes local topological structures from the neighborhood of the given vertex pairs, avoiding useless features. For further exploiting higher-order structural information, HELP also learns features from *hyper-substructure network* (HSN).Extensive experiments on six benchmark datasets have shown the state-of-the-art performance of HELP on link prediction.

CCS CONCEPTS

• **Information systems → Data mining**; • **Computing methodologies** → *Knowledge representation and reasoning*.

KEYWORDS

Link prediction, subgraph, graph classification, graph neural network, deep learning

ACM Reference Format:
Jian Zhang, Jun Zheng, Jinyin Chen, and Qi Xuan. 2020. Hyper-Substructure Enhanced Link Predictor. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3340531.3412096

1 INTRODUCTION

As a representative network analysis task, link prediction infers the linkage status of a given pair of vertices in a network. Due to its practicability, link prediction has been widely applied in various areas, such as commodity recommendation on e-commerce platforms and friends recommendation in online social networks (OSNs).

Heuristic link prediction approaches, such as *Common Neighbors* (CN), *Resource Allocation Index* (RA) and Katz Index, use either local or global similarity scores to make prediction [5]. For instance, in OSN like Weibo, candidate friends would be recommended according to the common friends one has with other users. Though straightforward and efficient, heuristic approaches only learn superficial structural features but fail to characterize the complexity of networks. Later developed network embedding algorithms, such as DeepWalk [6] and node2vec [2], focus on the contexts of vertices in the sequences generated by random walk or its variants. The random walk-based algorithms, however, need to learn embeddings over the whole network, which limits the scalability to large-scale networks even with parallelized computation. Apart from the need for plenty of predefined parameters, they are gradually exceeded by newly emerged graph neural networks (GNNs).

With *graph convolution network* (GCN) [4] as the representative, the newly developed GNNs have shown their power on many tasks. Despite the performance, most GNNs make inference over the whole network, which leads to high computational complexity and might bring in noise since not all vertices are useful for downstream tasks. Though graph attention network (GAT) [8] and GraphSAGE [3] try to learn embeddings over the neighborhood of vertices, they still focus on superficial features. And so does SEAL [10]. In social networks, a few individuals may consists of a group for some purposes. And similar cases like pair programming are also common in open source software development. The interactions between the groups could characterize the higher-order structural features of the network. However, few existing methods pay attention to such information.

To tackle the limitations of existing algorithms, we propose an end-to-end deep learning framework, namely *hyper-substructure enhanced link predictor* (HELP), for link prediction. Converting link prediction problem to graph classification, HELP infers linkage status of a given pair of vertices based on their neighborhood rather than the whole network. The neighborhood selected by personalized PageRank (PPR) [1] consists of the vertices closest to the given vertices, avoiding useless structural features. Neighborhood learning enables HELP with scalability to large-scale networks. And vertex feature vectors characterizing the relative position of the vertex in the network are constructed to improve the accuracy. To utilize higher-order topological structures, networks constructed by substructures of the subgraph, namely *hyper-substructure network* (HSN), are also used for link prediction. And we use 2^{nd}-order HSN to validate its advantage in this paper. The obtained networks are finally fed into a GNN to make prediction. The main contributions of the paper are summarized as follows:

- We propose an end-to-end deep learning framework, namely hyper-substructure enhanced link predictor (HELP), for link prediction. And its outstanding performance have been proved by extensive experiments.
- We convert link prediction problem to graph classification and infer linkage status based on subgraphs rather than the whole network, expending the scalability of HELP.
- We creatively introduce HSN into link prediction to utilize higher-order structural information, which offers insights of network structure mining.

2 PROBLEM DEFINITION

Suppose we have a graph $G = \langle V, E \rangle$ with a set of vertices $V = \{v_i | i = 0, 1, \cdots, N-1\}$ where N represents the number of vertices and a set of edges $E \subseteq V \times V$. Given a pair of vertices u and v, our goal is to infer the linkage status between u and v based on their neighborhood $\Gamma(u, v)$ and its hyper-substructure network.

3 METHOD

In this section, we give a detailed description of HELP in three parts: 1) neighborhood normalization, 2) HSN construction and 3) the deep learning-based link prediction framework.

3.1 Neighborhood Normalization

Before performing link prediction, we need to extract a subgraph of G for (u, v). Rather than directly using the 1^{st} or 2^{nd} neighbors of u and v which may cause the uncertainty of the subgraph size, we incorporate personalized PageRank (PPR) [1] into HELP to make subgraph extraction and normalization. And PPR is defined as

$$\Pi^{ppr} = \alpha(I_n - (1-\alpha)D^{-1}A)^{-1}, \tag{1}$$

where D is the degree matrix of G and α represents the restart probability. Each row $\pi(i) = \Pi^{ppr}_{(i)}$ is the PPR vector for vertex i and the element $\pi(i)_j$ reflects the closeness between vertex i and j. We then sort $\pi(i)$ in descending order of the element value and choose the first N_{nb} vertices to construct the subgraph which is denoted by $G(u, v)$. We make random selection if there exists multiple nodes of the same PPR probability. We obtain $\Gamma(u)$ and $\Gamma(v)$ for vertex u and v, respectively. The vertices in $\Gamma(u)$ and $\Gamma(v)$ are linked if they are connected in G. When $\Gamma(u)$ and $\Gamma(v)$ contain the same vertex, they will be linked but labeled as different vertices in $G(u, v)$. Also, we construct a feature matrix for $G(u, v)$. For vertex i in $G(u, v)$, the feature vector is defined as the concatenation of one-hot encoded $LSP(i, u)$ on $G(u, v)$, where $LSP(i, u)$ denotes the distance, i.e. length of shortest path, between i and u on $G(u, v)$.

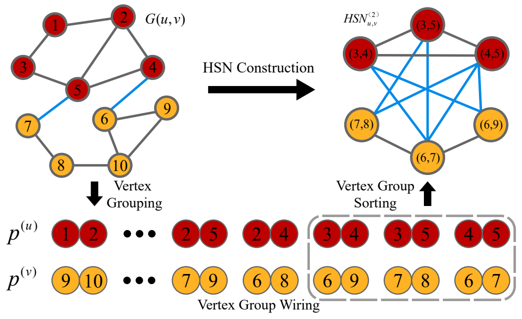

Figure 1: Illustration of $HSN^{(2)}_{u,v}$'s construction. The red vertices represent the vertices in $G(u)$ and those of yellow are the vertices in $G(v)$. The blue lines denote the interactions between $G(u)$ and $G(v)$.

3.2 HSN Construction

$G(u, v)$ characterizes the local structural features of u and v but do not describes the interactions between small groups in the network. To exploit higher-order structural information, we propose HSN to characterize the interactions of the substructures in $G(u, v)$. K^{th} order HSN, denoted by $HSN^{(K)}$, is constructed by following steps:

(1) **Vertex Grouping**. Group the vertices in $G(u)$ and make sure that each vertex should be grouped with arbitrary vertex at least once. The groups are represented by $p^{(u)} = \{p^{(u)}_1, \cdots, p^{(u)}_M\}$ where $M = C^K_{N_{nb}}$ and $p^{(u)}_i$ consists of K vertices randomly selected from $G(u)$. Also, we can obtain $p^{(v)}$ following the same procedure.

(2) **Vertex Group Sorting**. Sort $p^{(u)}$ in ascending order of distance $d(p^{(u)}_i | G(u, v))$ which is defined as

$$d(p^{(u)}_i | G(u, v)) = \sum_{j=0}^{K} LSP\left(p^{(u)}_i(j), v\right). \tag{2}$$

And we process $p^{(v)}$ in the same way. but with

$$d(p^{(v)}_i | G(u, v)) = \sum_{j=0}^{K} LSP\left(p^{(v)}_i(j), u\right). \tag{3}$$

(3) **Vertex Group Wiring**. After sorting, we select the first N_H groups from $p^{(u)}$ and $p^{(v)}$, respectively. Given $p^{(u)}_i$ and $p^{(u)}_j$, there exists an edge between them if $\left|p^{(u)}_i \cap p^{(u)}_j\right| < \left|p^{(u)}_i\right| + \left|p^{(u)}_j\right|$; Given $p^{(u)}_i$ and $p^{(v)}_i$, they are connected if there is at least one pair of vertices $\left(p^{(u)}_i(j_0), p^{(v)}_i(j_1)\right)$ is connected on $G(u, v)$.

Fig. 1 gives an example of $HSN^{(2)}_{u,v}$ construction with $N_{nb} = 5$, $K = 2$ and $N_H = 2$. The groups with small d, such as $(4, 5)$ and $(6, 7)$, are used for the construction of $HSN^{(2)}$, which enhances the interactions between $G(u)$ and $G(v)$.

3.3 The HELP Model

After obtaining the neighborhood of (u, v), we use an end-to-end deep learning framework, namely HELP, to infer whether there

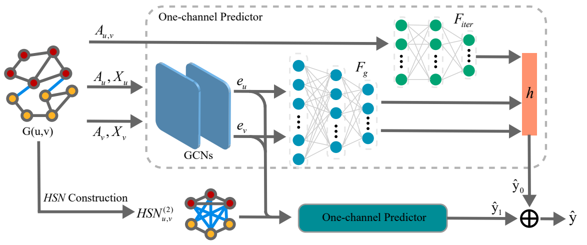

Figure 2: The framework of HELP.

exists an edge between u and v or not. We introduce graph convolution network (GCN) [4] into our framework to learning vertex embeddings. One layer GCN is defined as

$$GCN(A, X) = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}XW), \tag{4}$$

where $\tilde{A} = A + I$ and $\tilde{D}$ is the normalized degree matrix. W is the weight matrix and σ refers to the activation function. Here we use $\sigma \equiv ReLU(\cdot) = max(0, \cdot)$ in this paper.

The overall framework is given in Fig. 2. Given a network and a vertex pair (u, v), we process $G(u)$, $G(v)$ and the interactions between them separately, instead of processing $G(u, v)$ in its entirety. We design a deep learning-based model, called *one-channel predictor*, to deal with such input data. The *one-channel predictor* is described in Eq. (5):

$$\begin{aligned}
e_u &= GCNs(\tilde{A}_u, X_u),\ e_v = GCNs(\tilde{A}_v, X_v)\\
o^u &= FLATTEN(e_u),\ o^v = FLATTEN(e_v)\\
h_u &= F_g(e_u),\ h_v = F_g(e_v)\\
h_{u\times v} &= F_{iter}(A_{u\times v})\\
h &= CONCAT(h_u,\ h_v,\ h_{u\times v})\\
\hat{y}_0 &= softmax(W_{out}h + b),
\end{aligned} \tag{5}$$

where $\tilde{A}_u$ and X_u represent the normalized adjacency matrix and the feature matrix of $G(u)$, respectively. *One-channel predictor* first embeds the vertex in $G(u)$ and $G(v)$ with a 2-layer GCN separately and then encoded with a multi-layer perception F_g. As for e_u and e_v, they represent the vertex embeddings of $G(u)$ and $G(v)$, respectively. And o^u as the rows' concatenation of e_u denotes the embedding vector of $G(u)$ and so does o^v. The interaction between $G(u)$ and $G(v)$, denoted by $A_{u\times v}$, is also encoded by F_{iter}. The prediction result $\hat{y}_0$ is given by a softmax classifier. To integrate higher-order structural features, we also make prediction on $HSN^{(2)}_{u,v}$ with *one-channel predictor* and obtain the corresponding prediction result $\hat{y}_1$. The final result $\hat{y}$ is given by

$$\hat{y} = \frac{1}{2}(\hat{y}_0 + \hat{y}_1). \tag{6}$$

It's worth noticing that we use the concatenation of e_u and e_v generated based on $G(u, v)$ as the features when we make inference based on $HSN^{(2)}_{u,v}$. And the framework can be easily extended with higher-order HSN.

The objective function L_{total} mainly contains two parts: cross-entropy error L_c and embedding similarity error L_s. We minimize KL-divergence to measure the similarity of o^u and o^v if there exists an edge between u and v. L_{total} is define as

$$\begin{aligned}
L_c &= -y\log\hat{y} + (1 - y)\log(1 - \hat{y})\\
L_s &= \frac{1}{d}\sum_{i=0}^{d-1} y_i(o^u_i \log(o^u_i) - o^u_i \log(o^v_i))\\
L_{total} &= \gamma L_c + (1 - \gamma)L_s + \beta L_{reg},
\end{aligned} \tag{7}$$

where γ is the coefficient used for balancing L_c and L_s. L_{reg} represents L_2 regularization term and β is the weight decay coefficient.

Table 1: Basic statistics of the datasets.

	ATC	HP	NS	Cora	Citeseer	Power
N	1,226	1,706	1,589	2,708	3,279	4,941
$\lvert E\rvert$	2,615	6,207	2,742	5,278	4,552	6,594
Type	D	D	U	U	U	U

U means undirected network and D refers to directed network.

4 EXPERIMENT

4.1 Datasets

We evaluate the proposed method on six benchmark datasets:

- **ATC** is a directed network of US Air transportation where the vertices denote airports and the edges are flights.
- **HP** is a directed network of human protein interactions.
- **NS** is a scientist collaboration network where the vertices are scientists and the edges reflect co-authorship.
- **Cora** and **Citeseer** are networks modeling citation relationships between scientific papers.
- **Power** is an undirected network of power grid network where a vertex either represents a generator, a transformer or a substation and an edge is a power supply line.

The data are available online[1] and the basic statistics are summarized in Table 1.

4.2 Baselines

We compare the proposed method with five baselines, including random walking-based algorithms and deep learning-based models.

- **node2vec (N2V)** [2] performs biased random walk on networks and learns vertex embeddings through skip-gram.
- **SDNE** [9] embeds vertices into lower dimensional space in an auto-encoder-based framework.
- **SEAL** [10] is the abbreviation for *learning from Subgraphs, Embeddings and Attributes for Link prediction*. In this paper, we only focus on subgraphs but do not use other information.
- **GGAE** [7] with gravity-inspired decoder could reconstruct directed graphs from a node embedding.
- **SG** uses *one-channel predictor* to perform link prediction only based on $G(u, v)$.

4.3 Experimental Settings

In the experiments, each dataset is split into training and testing sets at the ratio of 4:1. Specifically, we choose 80% edges as positive samples of training set and randomly sample the same number

[1]http://konect.cc/

Table 2: Performance of link prediction (*AUC*)

Dataset	N2V	SDNE	SEAL	GGAE	SG	HELP
ATC	78.53	81.97	80.74	74.30	90.25	**91.13**
HP	**89.72**	85.86	88.94	88.56	87.93	89.52
NS	96.65	94.71	98.84	93.71	98.92	**99.12**
Power	59.20	58.07	83.99	74.67	86.03	**88.38**
Cora	72.26	69.96	91.81	84.64	91.67	**92.91**
Citeseer	69.93	71.09	87.54	86.38	85.88	**88.02**

Table 3: Performance of link prediction (*AP*)

Dataset	N2V	SDNE	SEAL	GGAE	SG	HELP
ATC	80.90	82.91	80.61	75.36	87.98	**88.84**
HP	90.42	86.77	89.22	89.17	90.13	**90.60**
NS	96.63	95.54	98.77	94.59	99.04	**99.27**
Power	59.21	60.71	86.03	79.49	87.29	**88.55**
Cora	72.41	73.46	92.98	86.66	92.99	**94.04**
Citeseer	69.64	74.97	88.74	87.47	86.12	**90.93**

of nonexistent training edges as negative training samples. The remaining 20% edges are positive testing data and the same amount of nonexistent edges are sampled as negative testing data.

As for the baselines, the embedding dimension of N2V is set to 128 and the optimal key parameters p and q are obtained through grid search over $\{0.50, 0.75, 1.00, 1.25, 1.50\}$. Also, the same embedding dimension is used when implementing SDNE. As for SEAL, we use the default settings and set the hop to *auto* which means 1-hop or 2-hop subgraphs will be automatically selected to achieve better performance. And GGAE also uses 128-dim latent vectors to represent nodes. For the proposed HELP, we set $N_{nb} = N_H = 35$ for subgraph and $HSN^{(2)}$ construction and train HELP for 1000 epochs at the learning rate of 3e-4. All the experiments are implemented on the platform equipped with a Intel(R) Xeon E5-2678 CPU and a NVIDIA GTX 1080Ti GPU.

4.4 Link prediction results

We employ *AUC* and *AP* as the metrics for evaluating the link prediction performance of HELP as well as the baselines. All experiments are conducted 10 times and average performance is reported to avoid contingency. As reported in Table 2 and Table 3, HELP outperforms other baselines in most cases considering both *AUC* and *AP*, which shows the effectiveness and practicability of HELP. Furthermore, N2V and SDNE have better performance on dense networks (HP and NS) rather than sparse networks (Citeseer and Power). SEAL, SG and HELP, however, behave relatively robust. It is due to that only a part of vertices, i.e. the neighborhood, could contribute to the prediction. N2V, SDNE and GGAE learn embeddings over the whole network are not always able to capture the key features and thus introduce noises into embedding vectors. SEAL, SG and HELP, however, make inference over the extracted subgraphs, preventing the model from noises. Compared with SG, the better performance of HELP validates that HSN could indeed help link prediction.

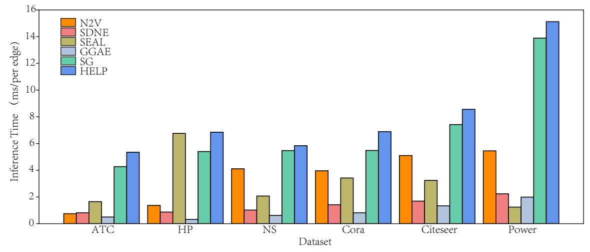

Figure 3: Inference time of HELP on different datasets.

4.5 Runtime Analysis

Above results have shown the effectiveness of HELP. And the scalability also matters. We show the inference time of a single vertex pair of HELP as well as the baselines in Fig. 3. HELP is relatively slower when making inference, while SDNE and GGAE are the most efficient among the 6 methods. And GGAE runs fastest since it makes prediction over the whole graph at once. Despite the inefficiency, each inference of HELP still could finish in milliseconds.

5 CONCLUSION

In this paper, we present hyper-substructure enhanced link predictor (HELP) which performs link prediction over the neighborhood of given vertex pair. Learning from subgraphs as well as their higher-order structural information modeled by hyper-substructure network (HSN), HELP outperforms other state-of-the-art baselines which have been proved by extensive experiments. Our future research will focus on optimizing the neighborhood normalization and HSN construction process to further compress the runtime without loss of accuracy.

ACKNOWLEDGMENTS

This work was partially supported by the National Natural Science Foundation of China under Grant 61973273, and by the Zhejiang Provincial Natural Science Foundation of China under Grant LR19F030001.

REFERENCES

[1] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Martin Blais, Amol Kapoor, Michal Lukasik, and Stephan Günnemann. 2019. Is PageRank All You Need for Scalable Graph Neural Networks?. In *Proceedings of the 15th MLG.*

[2] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd SIGKDD.* ACM, 855–864.

[3] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st NeuralPS.* 1024–1034.

[4] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of 5th ICLR.* OpenReview.net.

[5] Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications* 390, 6 (2011), 1150–1170.

[6] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th SIGKDD.* 701–710.

[7] Guillaume Salha, Stratis Limnios, Romain Hennequin, Viet-Anh Tran, and Michalis Vazirgiannis. 2019. Gravity-inspired graph autoencoders for directed link prediction. In *Proceedings of the 28th CIKM.* 589–598.

[8] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proceedings of the 6th ICLR.*

[9] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *Proceedings of the 22nd SIGKDD.* ACM, 1225–1234.

[10] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *Proceedings of 32nd NeurIPS.* 5171–5181.