CS230

Link Prediction with Graph Neural Networks and Knowledge Extraction

Zecheng Zhang
zecheng@stanford.edu

Danni Ma
dannima@stanford.edu

Xiaohan Li
xl1988@stanford.edu

Abstract

Link prediction is a core graph task by predicting the connection between two nodes based on node attributes. Many real-world tasks can be formed into this problem such as predicting academic article citations for specific topic. Recently, the advancement in graph neural network (GNN) has shifted the link prediction into neural style. Many GNN layers have been able to be applied to the link prediction task directly. But due to some graph structure and graph neural network limitations, the performance of the neural style link prediction sometimes will be negatively influenced. To address these issues, we propose a novel approach to implicitly guide GNN with extracted knowledge. The experiments on a biomedical dataset illustrates the state-of-the-art performance of our method.

1 Introduction

Many real-world tasks can be transformed into the graph related tasks, including the node classification, graph classification and link prediction etc. The link prediction is the crucial one. For example, predicting the citations among articles is usually transformed into the link prediction. The predicted citations will thus be helpful for further academic research. Recent development on graph neural network (GNN) provides the most promising approach to solve the link prediction task. Nevertheless, most GNNs learn the node embeddings through a relatively "shallow" neighborhood which contains information only two or three hops away from each node. In addition, in real-world link prediction tasks, the graphs are usually constructed through cut from a larger one or the nodes are manually selected from a specified pool. In the article citations prediction, articles are usually manually selected from specific topics. Those graphs usually suffer from some structural issues including the sparsity. Due to the shallowness of the neighborhood and those structural issues, the performance of GNNs on link prediction task will be negatively influenced.

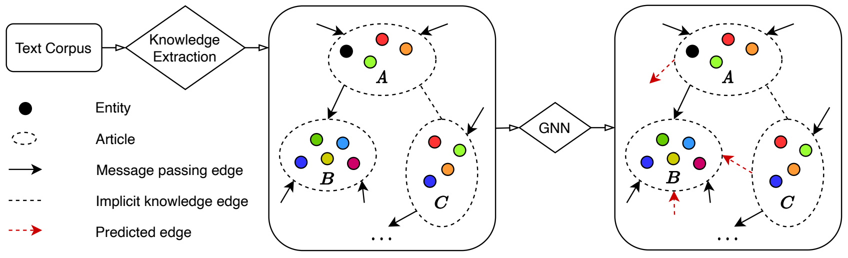


Figure 1: An overview of the general pipeline.

To better address the problems, we propose a knowledge driven approach to implicitly add edges before the training of GNNs. The method not only leverages the extracted node knowledge, but also utilizes samplings to improve the scalability. Extensive experiments show that our method can improve the link prediction results and we also evaluate the possible reasons why the prediction results are improved. The overall pipeline of our method is shown in Figure 1, and in the following sections we will describe on our method in details.

2 Related Work

Our method draws inspiration from graph neural networks, limitation of GNNs and the knowledge extraction. In this section, we will give a brief overview of these area.

Graph Neural Network. Graph neural network has been a popular research area for years. Recent advancement in graph neural networks offers the state-of-the-art learning ability on graph related tasks. GCN [6] utilizes spectral convolution to aggregate node features with respect to the local neighborhood. GraphSAGE [3] introduces a spatial aggregation of local node information by different aggregation ways. GAT [11] proposes an attention mechanism in the aggregation process by learning extra attention weights to the neighbors of each node.

Limitaton of Graph Neural Network. The number of GNN layers is limited due to the Laplacian smoothing [10]. Thus, the number of hidden layers in GNN usually is set to two or three. More layers will possibly result in the Laplacian smoothing result which will reduce the performance of learning. Recently a much deeper version of GCN [8] is proposed that the number of hidden layers can be added to hundred.

Knowledge Extraction. Knowledge extraction task in NLP has been extensively studied for decades. It includes many specific ones such as the named entity recognition (NER). Those extractions are important to many domains including the biomedical area [1]. The deep learning based NER identifies text spans into the named entities, and provides decent recognition performance [9]. Recent development of BERT [2] has instigated domain specific pretraining for NER such as the BioBERT [7] and BERN [4].

3 Real-world Link Prediction

3.1 Problem Statement

In real-world link prediction tasks, the graph G is usually a domain specific graph that each node contains information. For example, in the biomedical citation prediction task, the nodes are biomedical articles which have text information on genes, diseases and drugs. The link prediction task is that given the node features $\mathbf{X}$, the model can output whether two nodes are connected by an edge. To be more specific, GNN utilizes edges $E \in G$ (message-passing) to aggregate and learn node embeddings. The similarity score of two node embeddings will decide whether they should be connected. The errors (loss) on predictions will be backpropagated and updated the weights in neural networks.

3.2 Drawbacks

Graph Structure: Real-world graph G is usually cut from a larger network, and after outlier cleaning, it is usually very sparse and has many disconnected components. Those graph structural attributes can prevent the message-passing training for node embeddings and thus cause insufficient learning.

GNN Architecture: Mentioned in the related work section, usually GNNs are shallow. The limited number of layers prevents learning complex and high-level knowledge from the input features directly. Although some models can be stacked to hundred layers [8] in the point clouds dataset, but whether it works on smaller traditional link prediction task is still unstudied.

Node Feature: The input node features to the GNN usually are simple. Bag-of-word is often used if the node features are text. There are two reasons why choosing simple node features. First, using high-level node features can be expensive since we need to get the features to all the nodes. Second, the high-level node features might introduce extra noise which might even reduce model performance. But using only simple node features also limits the learning ability as the GNNs are often shallow.

4 LIKE: Link Prediction with Knowledge Extraction

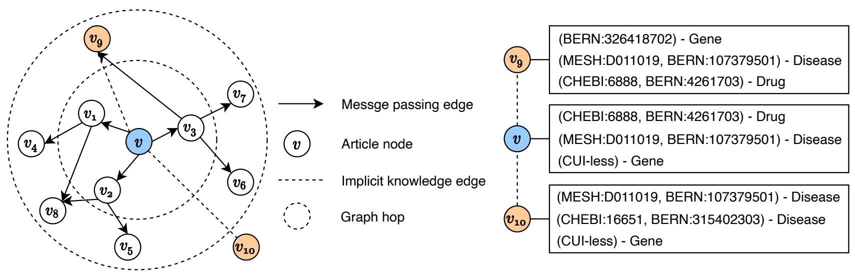
To address the drawbacks mentioned above, we propose a method called LIKE that the graph neural networks can be implicitly guided by high-level extracted knowledge.

4.1 Knowledge Extraction

The advanced knowledge extraction methods, such as BERN [4], can extract high quality knowledge. An NER example by BERN is shown in Figure 2. But as mentioned in the drawbacks section, it is expensive to extract high-level knowledge for all nodes. Thus, in LIKE we only extract a sampled fraction of nodes. The number of samples is shown below where N is the number of nodes in G:

$$N_s = \gamma \cdot N \text{ where } \gamma < 1 \tag{1}$$

Then we extract the knowledge for the sampled nodes, by using methods such as NER, which is



v_9
v_7
v_1
v_3
v_4
v
v_6
v_2
v_8
v_5
v_{10}
Messge passing edge
Article node
Implicit knowledge edge
Graph hop
(BERN:326418702) - Gene
(MESH:D011019, BERN:107379501) - Disease
(CHEBI:6888, BERN:4261703) - Drug
(CHEBI:6888, BERN:4261703) - Drug
(MESH:D011019, BERN:107379501) - Disease
(CUI-less) - Gene
(MESH:D011019, BERN:107379501) - Disease
(CHEBI:16651, BERN:315402303) - Disease
(CUI-less) - Gene

Figure 2: An illustration of the knowledge connected edge.

denoted as set S_s for N_s random nodes:

$$S_s = \{s_1, s_2, \cdots, s_{N_s}\} \tag{2}$$

Each item s_i is a set of knowledge attributes and lets assume s_i is associated with node $v_i \in V_s$, where V_s is the overall sampled node set. As shown in Figure 2, s_i can be set of entity tuples.

4.2 Implicit Knowledge Edge

As mentioned in previous section, there might be noise in high-level extracted knowledge features. So, they are not proper to be used as the input node feature directly. Also, the input requires consistency that all nodes should be extracted which will cost too much. Thus, we propose a concept of implicit knowledge edge to implicitly incorporate the high-level knowledge with similarity constraints.

We first set a threshold α and use Jaccard similarity to decide whether two sets s_i and s_j (for sampled nodes) are similar enough to be connected by an implicit knowledge edge. The set of implicit knowledge edges is added according to the equation below:

$$E_h = \bigcup_{v_i \in V_s} \bigcup_{v_j \in \mathcal{N}(\mathcal{N}(v_i))} \begin{cases} \{(v_i, v_j), (v_j, v_i)\}, & \text{if } \frac{|s_i \cap s_j|}{|s_i \cup s_j|} \geq \alpha \text{ and } (v_i, v_j) \notin E \text{ and } (v_j, v_i) \notin E \\ \emptyset, & \text{otherwise} \end{cases} \tag{3}$$

We set α to a relative high value to constrain the noise by making sure two nodes are very similar in high-level knowledge. We only compute the similarities among two-hop neighborhood, where the neighborhood of a node is denoted as $\mathcal{N}$, because comparing all pairs in S_s, the complexity is $O(N_s^2)$ which will still be very expensive. We argue that adding the set of E_h edges in message-passing will implicitly help the GNN learn the node embeddings through the constrained guide from extracted high-level knowledge. These guided and enriched message-passing edges will improve the model during the training.

Sometimes adding an edge into the graph might not help the GNN learn node embeddings much. One scenario is that when the node is a hub node, which has the number of edges much larger than that in average. Adding an edge in this case might cause the model learn something too common and somehow result in Laplacian smoothing effect. So, we add a filtering criteria to the nodes:

$$\mathbb{1}(v) = \begin{cases} 1, & \text{if } deg(v) \leq \beta \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

We notice that many components in G are disconnected due to the outlier removal and graph cut. Many of the components actually share some useful information that can be helpful in GNN learning. So, we sample r node pairs in V_s which can add implicit knowledge edges between different components:

$$E_r = \bigcup_{v_i \in V_s} \bigcup_{v_j \in V_s, v_j \neq v_i} \begin{cases} \{(v_i, v_j), (v_j, v_i)\}, & \text{if } \frac{|s_i \cap s_j|}{|s_i \cup s_j|} \geq \alpha \text{ and } (v_i, v_j) \notin E \text{ and } (v_j, v_i) \notin E \\ \emptyset, & \text{otherwise} \end{cases} \tag{5}$$

4.3 Combined with GNN

Then we combine the implicit knowledge edges with the training of GNN. Lets assume we have a one GNN layer $\mathcal{G}$, edges for loss computation E_{loss} a similarity function f. The forward process can be formulated as the following:

$$\mathbf{h} = \sigma(f(\mathcal{G}(\mathbf{X}, E \cup E_h \cup E_r)), E_{loss}) \tag{6}$$

The GNN layer $\mathcal{G}$ takes the node attributes $\mathbf{X}$ and edges $E \cup E_h \cup E_r$ for message-passing. The similarity function f takes the learned node embedding from the GNN layer $\mathcal{G}$ and loss computation edges E_{loss}. For each edge $e \in (E_{loss})$ where e connects node v_i and v_j, the f, usually cosine similarity, will compute the similarity score between them. At last, we will input the similarity score to an activation function σ, such as sigmoid, to get the result $\mathbf{h}$ in range of 0 to 1 and determine the connection by setting a threshold 0.5.

The layer in $\mathcal{G}$ is to aggregate information of nodes in the neighborhood by deep learning based methods. Some GNN layers are frequently used including the GraphSAGE layer (SAGE) [3] and graph attention layer (GAT) [11] etc.

5 Experiment

We conducted experiments[1] on a biomedical dataset to show the effectiveness of our approach.

5.1 Dataset

The dataset is from the COVID-19 Open Research Dataset Challenge[2]. It consists about 52000 manually selected scholarly articles with abstract corpus and references. After the cleaning, the data is formed into a graph with 27709 nodes (articles) and 84849 edges (citations). We use the 2842 dimension bag-of-word representation of the abstract as node features. The graph has a 3.062 average degrees and 981 weakly connected components, which is sparse and relatively disconnected.

5.2 Experiment Settings

Metrics: We use the accuracy and ROC-AUC score which are frequently used in link prediction.

Dataset: The dataset is splitted into three parts (train, validation and test) with ratio $85 : 5 : 10$. Since this is a binary classification task, we also sampled the same number of negative samples for each set. The negative samples are drawn disjointly, which means that the negative edges will not overlap with the positive edges and negative edges in other sets. The positive edges have labels 1 and negative ones have labels 0.

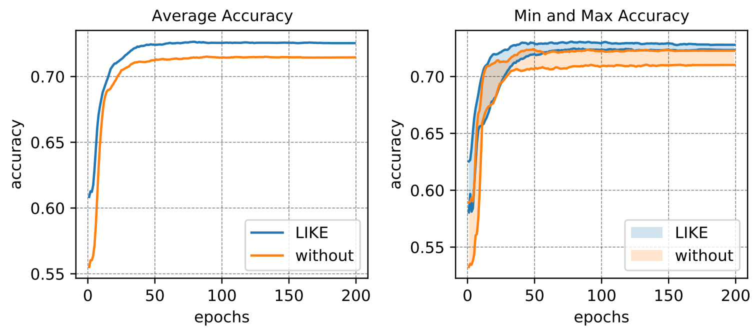
GNN Hyperparameter: The number of GNN layers is 2 with hidden size 16. Trained with Adam optimizer [5] with learning rate 0.001 and weight decay 5×10^{-4}.

GNN layer: The GNN layers include the SAGE [3] and GAT [11].

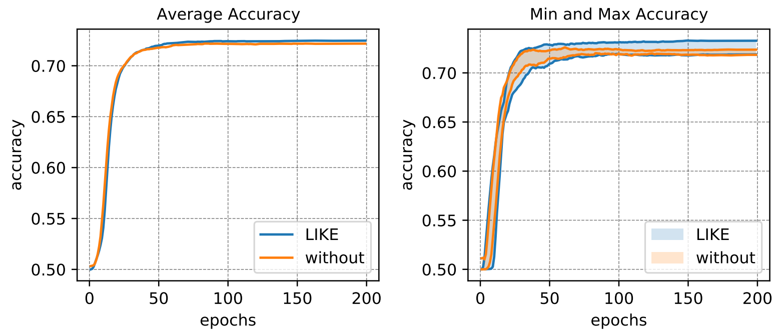
LIKE Hyperparameter: The hyperparameters might be different for different GNN layers. But the usual one is to set $\alpha = 0.7$, $\beta = 10$, $\gamma = \frac{1}{4}$ and $r = 20000$.

Knowledge Extraction: We use BERN [4] to extract named entities for the abstract of each articles.

[1]https://github.com/zechengz/gnn-ke
[2]https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge



(a) GAT



(b) GraphSAGE

Figure 3: Maximum, minimum and average test accuracy for each method by running 5 times.

5.3 Experiment Results

As shown in Table 1, the best performance of LIKE outperforms that of the models using only the original edges in message-passing. To further prove our method is better, we run each model 5 times and compare the maximum, minimum and average test accuracy. The results are shown in Figure 3.

-	GAT	GAT (LIKE)	SAGE	SAGE (LIKE)
Accuracy	72.01%	72.46% (+)	72.56%	**73.03**% (+)
ROC-AUC score	85.41%	**86.36**% (+)	85.39%	85.51% (+)

Table 1: Best link prediction results for each methods on test set.

5.4 Evaluation

We explored on the change of graph structures and try to find why our method can outperform the baselines. If we add 596 random implicit knowledge edges, set $\alpha = 0.7$ and $\beta = 10$, the LIKE will outperforms the original GNN. We analyzed the graph structure and found the graph connection in the message-passing stage will be somehow densified with the number of weakly connected components is reduced around 4% and the average degree of the graph increased by 0.1. We argure that the densification might be the reason that the learned node embeddings in GNN is improved.

6 Conclusion

In this project, we propose a novel idea to implicitly incorporate rich knowledge during the GNN message-passing in training. Because the limitations of the traditional GNN layers and the issues of the real-world link prediction graphs can negatively influence the performance, our implicit knowledge edge can help to reduce those problems. According to the experiments on a biomedical dataset, our method do help to better the overall performance in the real-world link prediction task.

7 Contribution

Zecheng Zhang wrote the code for link prediciton. Danni Ma helped on the knowledge extraction part and Xiaohan Li analyzed the graph structures. All of three contributed to the proposal writing, two milestone writings, method discussion and final report writing.

References

[1] Aaron M Cohen and William R Hersh. A survey of current work in biomedical text mining. *Briefings in bioinformatics*, 6(1):57–71, 2005.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[4] Donghyeon Kim, Jinhyuk Lee, Chan Ho So, Hwisang Jeon, Minbyul Jeong, Yonghwa Choi, Wonjin Yoon, Mujeen Sung, , and Jaewoo Kang. A neural named entity recognition and multi-type normalization tool for biomedical text mining. *IEEE Access*, 7:73729–73740, 2019.

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[7] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[8] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[9] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[10] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[11] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.