Structural Information Enhanced Graph Representation for Link Prediction

Lei Shi\textsuperscript{1}, Bin Hu\textsuperscript{1}, Deng Zhao\textsuperscript{1}, Jianshan He\textsuperscript{2}, Zhiqiang Zhang\textsuperscript{1}, Jun Zhou\textsuperscript{1}

\textsuperscript{1}Machine Intelligence Department, Ant Group,
\textsuperscript{2}Consumer Finance Technology Department, Ant Group

\{sl316606, axel hb, zhaodeng zd, yebai hjs\}@antgroup.com, \{lingyao zzq, jun zhouchun\}@antfin.com

Abstract

Link prediction is a fundamental task of graph machine learning, and Graph Neural Network (GNN) based methods have become the mainstream approach due to their good performance. However, the typical practice learns node representations through neighborhood aggregation, lacking awareness of the structural relationships between target nodes. Recently, some methods have attempted to address this issue by node labeling tricks. However, they still rely on the node-centric neighborhood message passing of GNNs, which we believe involves two limitations in terms of information perception and transmission for link prediction. First, it cannot perceive long-range structural information due to the restricted receptive fields. Second, there may be information loss of node-centric model on link-centric task. In addition, we empirically find that the neighbor node features could introduce noise for link prediction. To address these issues, we propose a structural information enhanced link prediction framework, which involves removing the neighbor node features while fitting neighborhood graph structures more focused through GNN. Furthermore, we introduce Binary Structural Transformer (BST) to encode the structural relationships between target nodes, complementing the deficiency of GNN. Our approach achieves remarkable results on multiple popular benchmarks, including ranking first on ogbl ppa, ogbl citation2 and Pubmed.

Introduction

Link prediction is the task of predicting missing or potential edges according to existing nodes and edges in a graph, the typical applications of which include knowledge completion, recommendation, and more. For example, it can complete missing colleague relationships, and city of birth for a person, or can predict users’ interest in certain products in a recommendation system. There are various methods for link prediction, among which Graph Neural Network (GNN) based methods have become the mainstream ones due to their good performance and efficiency.

GNNs (Kipf and Welling 2016a; Hamilton, Ying, and Leskovec 2017; Velikovi et al. 2017) have achieved great results in a lot of fields. They are centered on a certain node, and aggregate features of neighbor nodes and edges to the central node through the Message Passing (MP) (Gilmer et al. 2017), which are good at learning node representations. When applied to link prediction, the typical practice is to use the Autoencoder framework on Graph (GAE (Kipf and Welling 2016b)). This involves encoding individual node representations with GNNs, aggregating representations of the target nodes, and decoding them into link probability.

However, such an approach may miss some important features for link prediction. A simple example is illustrated in Fig. 1-(a), where nodes $h_1$ and $h_2$ are isomorphic. Without node features, GNN learns the same node representation, resulting in the same link representation of pair $(h_1, t)$ and pair $(h_2, t)$. In fact, the results should not be the same. Node $h_2$ and $t$ share a common neighbor, indicating a higher probability of a link. We believe that GAEs fail fundamentally because they focus on capturing neighbor information centered on nodes during the encoding stage, without perceiving the structural relationships of target node pairs.

Recently, some works have attempted to address this issue by node labeling tricks. For example, ID-GNN (You et al. 2021) colors target nodes to distinguish them from neighbor nodes, and SEAL (Zhang and Chen 2018; Zhang et al. 2021) calculates DRNL (Double-Radius Node Labeling) of all nodes in an enclosing subgraph. However, these approaches only help perceive path depth, while lacking perception of path ‘width’, such as node degree or the number of paths. Moreover, they are still limited by the MP paradigm of GNNs, leading to shortcomings in terms of information perception and transmission for link prediction. On one hand, the way of neighbor node labeling converging to target nodes and then aggregating into relationship representations is implicit and convoluted, potentially risking information loss. On the other hand, the receptive field of GNNs may not be sufficient enough for link prediction.

GNNs are node-centric models, whose receptive field is the neighborhood of a central node. However, some target nodes in link prediction may be far apart, and GNNs struggle to capture the long-range information between them. The right part of Fig. 1 shows a simple example, where neither ID-GNN nor SEAL with the most commonly used 3-layer GNNs can distinguish between the two link structures shown in (b) and (c). When computing the representation of the head nodes $h$ using GNNs, the computational graphs (including nodes’ DRNL) in both structures are the same, as
shown in (d), and the color of tail nodes $t$ are not aggregated. The same applies when computing the representation of the tail nodes $t$. Therefore, the two link structures cannot be distinguished by 3-layer GNNs, even with labeling tricks.

To enhance the long-range information capturing ability of GNNs, the number of GNN layers needs to be increased to expand the receptive field, which, however, may lead to problems such as oversmoothing and overfitting (Li, Han, and Wu 2018; Rong et al. 2020).

In this paper, we propose an additional attention module to compensate for the lack of GNN in capturing pairwise structures and directly learn relationship representations between target nodes. Self-attention is a powerful mechanism to learn pairwise relationships and shines brightly in Transformers (Vaswani et al. 2017), but it lacks the ability to perceive structures of elements. In addition, the computation is performed pairwise between all elements, resulting in a quadratic computational complexity and a large number of parameters leading to the problem of data hunger (Hassani et al. 2021). In order to capture the pairwise structural features of target nodes, we introduce the topological encoding to encode pairwise heuristics such as shortest path distance (SPD), Adamic-Adar, etc., and fuse them into attention. Besides, our encoding target is not the global information, but rather the pairwise structures of target nodes missed by GNNs. Therefore, we restrict attention computation to the target nodes, avoiding expensive computations and the risk of introducing noise from other nodes. As a result, the proposed attention module focuses on the target nodes and effectively encodes structural features, which we refer to as Binary Structural Attention (BSA).

On the other hand, GNNs aggregate rich information from the neighborhood, mainly including graph structure information and the features of neighbors. However, a few works have raised doubts about neighbors in graph learning, which prompt us to question the necessity of neighbor features in link prediction. Thus, we attempted to remove neighbor node features. Taking SEAL (with NGNN on ogbl-ppa and ogbl-citation2) as an example in Table 1, we observed the improvement without neighbor node features (experimental settings are the same as the proposed method, described in the Experiments section and Appendix). This phenomenon contradicts intuition yet seems to be explainable: in link prediction, neighbor node features may be task-irrelevant and prone to noise. For example, in predicting a spouse relationship, the key information could be that two individuals have a common child, but the child’s interests, phone number, and other features may not be important.

Based on the above, we propose a Structural Information Enhanced Graph representation framework for link prediction (SIEG). On one hand, we use a GNN to collect the neighborhood information of target nodes, removing neighbor node features to make the model more focused on structural features. On the other hand, the BSA is introduced to encode the pairwise features of the target nodes, especially the long-range structural features, to complement the deficiency of GNN. Our method shows remarkable performance on six popular benchmarks, including ranking first on ogbl-ppa, ogbl-citation2 and Pubmed.

### Related Works

**Link Prediction.** There are four main paradigms of existing works on link prediction: walk-based methods, shallow Knowledge Graph Embedding methods (shallow-KGEs), path-based methods, and GNN-based methods. Walk-based methods take the random walk sequence from a node as the representation of it, whose representatives are DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) and Node2Vec (Grover and Leskovec 2016). Shallow-KGEs focus more on the triple relationship of head node, edge, and tail node, in-
cluding TransE (Bordes et al. 2013), DistMult (Yang et al. 2014), and subsequent works (Trouillon et al. 2016; Sun et al. 2019; Zhang et al. 2020b) improving score functions. However, these methods lack inductive properties and parameter sharing mechanisms, making them difficult to apply to large and scalable graphs. Path-based methods focus on connectivity between target nodes, including heuristic methods such as SPD and Katz (Katz 1953), rule-based approaches like NeurallLP (Yang, Yang, and Cohen 2017) and DRUM (Sadeghian et al. 2019), path representation methods such as Path-RNN (Neelakantan, Roth, and McCallum 2015) and others (Wang, Ren, and Leskovec 2020; Zhu et al. 2021; Kong, Chen, and Zhang 2023). Among them, the heuristic methods are simple, interpretable, and inductive, making them suitable as structural features in our BSA.

Due to good performance and efficiency, GNN-based methods have become the mainstream approach for link prediction. Prevalent methods (Kipf and Welling 2016b; Vashishth et al. 2019; Ahn and Kim 2021) adopt an Autoencoder framework, where GNN serves as the encoder for node representations and edges are decoded as a function over node pairs. Another stream of frameworks, including SEAL (Zhang and Chen 2018), IGMC (Zhang and Chen 2020), and GraIL (Teru, Denis, and Hamilton 2020), encodes the enclosing subgraph around each node pair for link prediction. However, a good node-level representation does not necessarily lead to a good link-level representation (Zhang et al. 2021). Methods (You et al. 2021; Zhang et al. 2021) bridges the gap between node-level and edge-level GNN expressiveness with labeling tricks. However, they are still constrained by the MP paradigm of GNNs, unable to capture comprehensive long-range structures directly.

**Graph Transformer.** Transformer has achieved great success in Natural Language Processing (Vaswani et al. 2017) and Computer Vision (Dosovitskiy et al. 2021), and its application to graph has also emerged. The Transformer faces two main challenges when applied to graph data: position/structure encoding and computational complexity. Firstly, the core module of Transformer, Self-attention, lacks the ability to perceive the structural information of targets. To address this, some methods have introduced additional encoding schemes, such as Laplacian encoding (Dwivedi and Bresson 2020), distance correlation encoding (Ying et al. 2021; Mialon et al. 2021), and structure-aware encoding (Chen, O’Bray, and Borgwardt 2022; Park et al. 2022), while some works propose specialized networks to learn positional encoding (Kreuzer et al. 2021). In order to capture the pairwise structural features of target nodes in BSA, we extend the topological encoding (Park et al. 2022) to encode pairwise heuristics and fuse them into attention.

In addition, Self-attention calculates the similarity between all nodes pairwise, resulting in a quadratic computational complexity. To alleviate this problem, methods (Dwivedi and Bresson 2020; Zhang et al. 2020a) limit the number of nodes involved in computation through sampling. However, the sampling strategies are still node-centric rather than link-centric. For BSA, the encoding target is specifically the pairwise structures of target nodes in link prediction. Therefore, we restrict attention computation to the target nodes, avoiding the quadratic computations.

**Doubt Neighbors.** A few works have raised doubts about neighbors in graph learning. Feng et al. (Feng et al. 2021) suggests that some of the neighbors in GNN aggregation may be unreliable and misleading. While GraphENS (Park, Song, and Yang 2022) have found that models tend to overfit to neighbor sets of minor class in class-imbalanced node classification. However, there is no research questioning the necessity of neighbor node features in link prediction, to the best of our knowledge.

**Methodology.**

**Problem Definition.** Assuming a graph is denoted as \( G = (V, E, A) \), where \( V \), \( E \) are the set of nodes and the set of edges, respectively, and \( A \) is the adjacency matrix of the graph. Node feature matrix is denoted as \( X \in \mathbb{R}^{V \times D} \), where \( D \) is the dimension of node features. The goal of link prediction is to predict whether there exists an edge \( e_{i,j} \) between two given nodes \((v_i, v_j)\), according to the graph.

**Overall Architecture.** As it is shown in Fig. 2, the proposed link prediction framework consists of four main parts: subgraph sampling, structural information enhanced GNN for neighborhood structure encoding, Binary Structural Transformer (BST) for pairwise structure encoding, and link decoding.

**Subgraph Sampling.**

Given a target node pair \((v_i, v_j)\), we sample a k-hop enclosing subgraph (Zhang and Chen 2017) around it. This subgraph is composed of the union of the two k-hop subgraphs around each target node, with edges between the two subgraphs also being incorporated. For instance, in Fig. 2, an additional edge \( e_{1,5} \) beyond 1-hop is included in the 1-hop enclosing subgraph of target node pair \((h, t)\). Enclosing subgraphs provide better connectivity and more informative input for link prediction than normal subgraphs. It has been proved in WLNM (Zhang and Chen 2017) that not only first and second-order heuristic features can be directly calculated in enclosing subgraph, but even high-order heuristics are also feasible with a small number of hops.

**Neighborhood Structure Encoding.**

To encode the neighborhood structures and learn the representations of each target node, any GNN can be used. GNNs aggregate neighborhood information through MP centered around nodes, and the typical form (Kipf and Welling 2016a) is shown as

\[
H^{(l+1)} = \sigma(\tilde{A}H^{(l)}W^{(l)}),
\]

(1)

Where \( \tilde{A} := \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2} \) denotes the normalized adjacency matrix, \( \tilde{A} := A + I \), where \( I \) is identity matrix. \( \tilde{D}_{ii} := \sum_j \tilde{A}_{ij} \). \( H^{(l)} \) is the node embedding matrix and \( W^{(l)} \) is the linear parameter matrix in the \( l \)-th layer. \( \sigma \) represents the ReLU activation function. Typically, similarity representations can be obtained by aggregating the respective embeddings of the target nodes. In this work, we adopt
the SortPooling (Zhang et al. 2018) to form the similarity representation.

To enhance the fitting to structural information, we remove the neighbor node features to avoid potential noise, making the model focus more on the neighborhood structures of target nodes. Specifically, the information inputted to the GNN does not include any node features, but instead adopts the structural features encoded by DRNL (Zhang and Chen 2018). DRNL encodes the distances between each node within the subgraph and the target nodes, the specific form of which is shown as

$$f_l(i) = \begin{cases} 1 + \min(d_x, d_y) + (d/2)|\lfloor (d/2) \rfloor + (d/2)-1, & \text{if } d_x \cdot d_y \neq 0, \\ 1, & \text{otherwise.} \end{cases}$$

(2)

Here, $d_x$ and $d_y$ are the distances from a node to the two target nodes, respectively. $d := d_x + d_y$, $(d/2)$ and $(d/2)-1$ correspond to the integer quotient and remainder of $d$ divided by 2, respectively. On the other hand, the node features of target nodes is encoded with the additional Feed-forward Network (FFN), and then concatenated with the structural representations encoded by GNN before pooling.

**Pairwise Structure Encoding**

As we analyzed in the Introduction section, even with labeling tricks, the GNN framework has two limitations for link prediction: the absence of long-range structural information due to restricted receptive fields and the conflict between node-centric models and link-centric tasks. To address these shortcomings, we propose the BST to capture pairwise structures and feature similarities between target nodes exclusively, representing relations directly and supplementing the limitations of GNNs.

The calculation of BSA, which forms the core of BST, is illustrated on the right side of Fig. 2. The process involves two inputs: the features of the target nodes and the structural features of the target node pair. The latter are calculated using heuristic methods such as shortest path distance (SPD), number of shortest paths (SPN), common neighbor (CN), Jaccard index, Adamic-Adar index (AA), and so on. For instance, the Jaccard and AA are computed as

$$S_{\text{Jaccard}}[u, v] = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|},$$

(3)

$$S_{\text{AA}}[u, v] = \frac{1}{\log(d_z)},$$

(4)

where $u$ and $v$ denote the target nodes. $N(\cdot)$ denotes the set of neighbor nodes, and $|\cdot|$ denotes the number of elements in the set. $\cap$ and $\cup$ denote intersection and union of sets, respectively. $d_z$ denotes the degree of the common neighbor $z$. In addition to describing the connectivity path of the node pair, these two heuristics also introduce number of edges of target nodes and degree of the common neighbor, respectively, for normalization. Furthermore, we adopt structural lookup embedding in conjunction with node feature embeddings to obtain the structural encoding, thereby enhancing the representation of structure information:

$$s_{ij} = q_i \cdot \mathcal{P}_{\psi(i,j)}^{\text{query}} + k_j \cdot \mathcal{P}_{\psi(i,j)}^{\text{key}},$$

(5)
where \( q_i = h_i W^Q \), \( k_j = h_j W^K \).

\[
p^\text{query}_{\psi(i,j)} = p^\text{query}_{\text{SPD}(i,j)} + p^\text{query}_{\text{Jaccard}(i,j)} + p^\text{query}_{\text{AA}(i,j)} + \cdots \tag{7}
\]

\[
p^\text{key}_{\psi(i,j)} = p^\text{key}_{\text{SPD}(i,j)} + p^\text{key}_{\text{Jaccard}(i,j)} + p^\text{key}_{\text{AA}(i,j)} + \cdots \tag{8}
\]

Here, \( p^\text{query}_{\text{SPD}(i,j)} \) and \( p^\text{key}_{\text{SPD}(i,j)} \) denote two structural vectors obtained by using SPD as an index to lookup two separate parameterized tables, \( p^\text{query}_{\psi(i,j)} \) and \( p^\text{key}_{\psi(i,j)} \). Structural vectors obtained from other heuristics follow the same process. Next, we aggregate all the query structural vectors as \( p^\text{query}_{\psi(i,j)} \), and all the key structural vectors as \( p^\text{key}_{\psi(i,j)} \). Then, we multiply the two structural vectors with \( q_i \) and \( k_j \) respectively, and add them to obtain the structural encoding, denoted as \( b_{ij} \). Here, \( q_i \) and \( k_j \) are linear transformations of the target node features, with \( W^Q \) and \( W^K \) representing the transformation matrices. This effectively fuses the rich heuristic structures with node features, and characterizes the structural correlation of target nodes. Finally, the structural encoding \( b_{ij} \) is incorporated into attention coefficient \( a_{ij} \):

\[
a_{ij} = \frac{q_i \cdot k_j^\top + b_{ij}}{\sqrt{d}} \tag{9}
\]

Where \( d \) is the dimension of \( q_i \). To obtain the result of a single-head BSA, we multiply the softmax-normalized attention coefficient with the Value embedding, and we concatenate multiple heads to form the multi-head BSA:

\[
\text{Attn}_k(h_i, h_j) = \text{softmax}(a_{ij}) \cdot v_j \tag{10}
\]

\[
\text{Attn}(h_i, h_j) = \left[ \sum_{k=1}^K \text{Attn}_k(h_i, h_j) \right] W^O \tag{11}
\]

Where the Value embedding, \( v_j := h_j W^V \), is the linear transformation of \( h_j \) through \( W^V \). The symbol \( \| \) represents the concatenate operation, and \( W^O \) is a linear transformation matrix for multi-head fusion.

Lastly, the BST block comprises of the BSA, Layer Normalization (LN), Residual Connection (He et al. 2015), and FFN components, as depicted in Fig. 2. Similar to the regular Transformer, the pairwise structure encoding module, BST, is obtained by stacking multiple BST blocks.

The proposed BST offers several advantages over the regular Transformer. Firstly, the potential noise and overfitting risk introduced by nodes away from the targets are avoided. Secondly, more task-relevant structure information is collected. Finally, computational complexity is greatly reduced.

### Computational Complexity

In comparison to FCA, BSA achieves a significant reduction in computational complexity. FCA entails two main computational costs: SPD searching and Attention calculation, with complexities of \( \mathcal{O}(N(N+E)\log N) \) (Dijkstra 1959) and \( \mathcal{O}(N^2D+ND^2) \), respectively. Here, \( N \) and \( E \) denote the number of nodes and edges in the subgraph, and \( D \) denotes the dimension of node features. In contrast, the computational complexity of BST is much lower than that of FCA, as indicated in Table 2, with only \( \mathcal{O}((N+E)\log N) \) for SPD and \( \mathcal{O}(D^2) \) for Attention.

The overall computational cost of the Transformer is proportional to the sum of the computational cost of SPD and Attention. Consequently, the acceleration ratio from fully-connected Transformer (FCT) to BST ranges from \( N \) to \( N^2 \), which can be approximated to \( N \) or \( N^2 \) when \( N \ll D \) or \( N \gg D \) respectively. BST effectively addresses the computationally expensive nature of the Transformer in link prediction, enabling SIEG to be applied on larger-scale datasets.

### Expressive Power

The Weisfeiler-Lehman test (Weisfeiler and Leman 1968) is a widely used graph isomorphism test, and it has been proven (Xu et al. 2018) that the expressive power of Message Passing GNNs (MPGNNs), such as GCN, GraphSAGE, and GAT, is upper-bounded by the 1-dimensional Weisfeiler-Lehman (1-WL) algorithm. Specifically, the 1-WL algorithm cannot distinguish nodes with the same subtrees but different substructures. Consequently, the GAEs based on MPGNNs cannot distinguish link structures involving these target nodes. In contrast, the proposed SIEG is able to effectively distinguish such structures by providing distinct representations, even beyond receptive fields of GNNs. This implies that SIEG surpasses the 1-WL test and overcomes the message passing limitation of GNNs. The detailed proof is presented in the Appendix.

### Experiments

#### Datasets and Tasks

In order to prove the effectiveness of the proposed method, we conduct link prediction tasks on six popular datasets: three Open Graph Benchmark (OGB) datasets (Hu et al. 2020) including ogbl-ppa, ogbl-citation2, ogbl-vessel, and three classic attributed graph datasets (categorized as Classic in this paper) including Pubmed (Namata et al. 2012), Cora (McCallum et al. 2000), Citeseer (Giles, Bollacker, and Lawrence 1998). The details of the datasets, statistics, specific prediction tasks, and evaluation metrics are presented in the Appendix.
### Baselines
As the six datasets are widely popular, abundant baseline methods have been evaluated on them. On the OGB benchmarks, we classify the main baselines into three categories: heuristic methods including common neighbor (CN) (Liben-Nowell and Kleinberg 2007), Adamic-Adar (AA) (Lada et al. 2003), resource allocation (RA) (Zhou and Zhang 2009); non-GNN learning methods including Node2vec (Grover and Leskovec 2016), Matrix Factorization (MF) (Koren, Bell, and Volinsky 2009), MLP; GNN-based methods including GCN (Kipf and Welling 2016a), GraphSAGE (Hamilton, Ying, and Leskovec 2017), SEAL (Tan et al. 2023), AGDN (Sun et al. 2022), S3GRL (Louis, Jacob, and Salehi-Abari 2023), Neural CN (Wang, Yang, and Zhang 2023), SUREL (Yin et al. 2023). On the Classic datasets, we classify the main baselines into three categories (the citations are the experimental results sources): heuristic methods including CN (Louis, Jacob, and Salehi-Abari 2023), AA (Louis, Jacob, and Salehi-Abari 2023), Personalized PageRank (PPR) (Louis, Jacob, and Salehi-Abari 2023), Katz (Zhu et al. 2018), Adamic-Adar (AA) (Lada et al. 2003), resource allocation (RA) (Zhou et al. 2009); non-GNN learning methods including common neighbor (CN) (Liben-Nowell and Kleinberg 2007), Adamic-Adar (AA) (Lada et al. 2003), resource allocation (RA) (Zhou et al. 2009); non-GNN learning methods including common neighbor (CN) (Liben-Nowell and Kleinberg 2007), Adamic-Adar (AA) (Lada et al. 2003), resource allocation (RA) (Zhou et al. 2009); non-GNN learning methods including common neighbor (CN) (Liben-Nowell and Kleinberg 2007), Adamic-Adar (AA) (Lada et al. 2003), resource allocation (RA) (Zhou et al. 2009); non-GNN learning methods including common neighbor (CN) (Liben-Nowell and Kleinberg 2007), Adamic-Adar (AA) (Lada et al. 2003), resource allocation (RA) (Zhou et al. 2009); non-GNN learning methods including common neighbor (CN) (Liben-Nowell and Kleinberg 2007), Adamic-Adar (AA) (Lada et al. 2003), resource allocation (RA) (Zhou et al. 2009); non-GNN learning methods including common neighbor (CN) (Liben-Nowell and Kleinberg 2007), Adamic-Adar (AA) (Lada et al. 2003), resource allocation (RA) (Zhou et al. 2009); non-GNN learning methods including common neighbor (CN) (Liben-Nowell and Kleinberg 2007), Adamic-Adar (AA) (Lada et al. 2003), resource allocation (RA) (Zhou et al. 2009); non-GNN learning methods including common neighbor (CN) (Liben-Nowell and Kleinberg 2007), Adamic-Adar (AA) (Lada et al. 2003), resource allocation (RA) (Zhou et al. 2009); non-GNN learning methods including common neighbor (CN) (Liben-Nowell and Kleinberg 2007), Adamic-Adar (AA) (Lada et al. 2003), resource allocation (RA) (Zhou et al. 2009); non-GNN learning methods including common neighbor (CN) (Liben-Nowell and Kleinberg 2007), Adami...
Table 5: Performance of SIEG and its variants.

<table>
<thead>
<tr>
<th>Category</th>
<th>Model</th>
<th>ogbl-ppa Hits@100 (%)</th>
<th>ogbl-ppa MRR (%)</th>
<th>ogbl-citation2 Hits@100 (%)</th>
<th>ogbl-citation2 MRR (%)</th>
<th>ogbl-vessel Hits@100 (%)</th>
<th>ogbl-vessel MRR (%)</th>
<th>Pubmed AUC (%)</th>
<th>Cora AUC (%)</th>
<th>Citeseer AUC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIEG</td>
<td>w/o NF</td>
<td>63.22±1.74</td>
<td>89.87±0.18</td>
<td>96.87±0.01</td>
<td>98.74±0.04</td>
<td>93.92±0.05</td>
<td>92.10±0.44</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>w/ NF</td>
<td>61.88±4.33</td>
<td>89.57±0.10</td>
<td>96.77±0.01</td>
<td>98.50±0.09</td>
<td>93.04±0.54</td>
<td>90.85±0.74</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>w/o BST</td>
<td>60.05±4.85</td>
<td>89.18±0.11</td>
<td>84.70±0.70</td>
<td>97.94±0.08</td>
<td>93.24±0.43</td>
<td>88.22±1.18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>w/o GNN</td>
<td>33.46±2.58</td>
<td>85.86±0.04</td>
<td>94.88±0.01</td>
<td>96.32±0.23</td>
<td>91.26±0.47</td>
<td>85.57±1.29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIEG (FCT)</td>
<td>w/o NF</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>97.35±0.31</td>
<td>93.22±0.33</td>
<td>86.94±1.98</td>
<td></td>
</tr>
<tr>
<td></td>
<td>w/ NF</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>96.55±0.35</td>
<td>92.30±0.46</td>
<td>84.69±0.90</td>
<td></td>
</tr>
<tr>
<td></td>
<td>w/o GNN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>95.56±0.13</td>
<td>90.35±0.60</td>
<td>84.03±0.16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>w/o FCT</td>
<td>60.05±4.85</td>
<td>89.18±0.11</td>
<td>84.70±0.70</td>
<td>97.94±0.08</td>
<td>93.24±0.43</td>
<td>88.22±1.18</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Primarily, we analyze intra-category results based on SIEG w/o NF. Adding neighbor features (SIEG w/ NF) leads to worse performance, consistent with the results of GNN alone in Table 1. This confirms our suspicion that neighbor features are not effective features for link prediction or introduce noise. Besides, removing BST (SIEG w/o BST) leads to a drop in performance, indicating that BST learns important features missed by GNN, particularly pairwise structural features. In SIEG (FCT) category, adding neighbor features and removing GNN based on SIEG (FCT) w/o NF lead to drops, while removing FCT leads to a rise. This suggests that FCT negatively impacts link prediction on top of GNN, despite capturing pairwise structures, like BST. What causes this damage?

Here, we analyze the cross-category results by comparing each model of SIEG with the corresponding model of SIEG (FCT). We find that replacing BST with FCT results in a drop in all models, which supports the conclusion analyzed in the Methodology section. FCT introduces a large amount information away from the targets, which may lead to potential noise or overfitting risks.

**Computational Efficiency.** We present the Time consumption results of SIEG and SIEG (FCT) on a small dataset (Cora) and a medium-size dataset (ogbl-citation2) in Fig. 3. As theoretically analyzed in the Methodology section, the computational cost ratio between FCT and BST ranges from $N$ to $N^2$. Experimentally, the results on Cora show that SIEG w/o NF runs 10 times faster than SIEG (FCT) w/o NF, and SIEG w/o GNN runs 16 times faster than SIEG (FCT) w/o GNN, demonstrating the significant efficiency advantage of BST over FCT. When dealing with medium-sized ogbl-citation2, models with FCT can hardly run.

**Case Study.** We use a simple example to illustrate how SIEG leverages pairwise structural features for link prediction. On Cora test set, SIEG assigns a high score of 8.22 to a positive publication pair with target node IDs (1111, 1273), and predicts a citation relationship between them. Examination of the pairwise structural features of this pair reveals a CN score of 4 and an AA score of 2.21, indicating the presence of 4 common citations who are not widely cited, which should be an important reason for SIEG to infer their citation relationship.

**Conclusions**

In this paper, we theoretically analyze GNNs’ two inherent defects for link prediction in terms of information perception and transmission, and experimentally discover the harmful effects of neighbor node features. To address these issues, we propose a structural information enhanced link prediction framework, which involves removing the neighbor node features while fitting neighborhood structures more focused through GNN, and introducing BST to encode the structural relationships between target nodes, complementing the defects of GNN. The proposed method achieves remarkable results on six popular benchmarks, including ranking first on ogbl-ppa, ogbl-citation2 and Pubmed. Lastly, note that link prediction is a special case of multi-node representation learning, which also includes triplet (Liu, Ma, and Li 2021), motif (Besta et al. 2021), and subgraph (Alsentzer et al. 2020) tasks, among others. Theoretically, our method is also applicable to these tasks. In future work, we will practically extend SIEG to multiple tasks and develop it into a general multi-node representation learning framework.
References


Dwivedi, V. P.; and Bresson, X. 2020. A Generalization of Transformer Networks to Graphs.


Grover, A.; and Leskovec, J. 2016. node2vec: Scalable Feature Learning for Networks. ACM.


Kipf, T. N.; and Welling, M. 2016b. Variational Graph Auto-Encoders.


Lada: A; Adamic; ; ; Eytan; and Adar. 2003. Friends and neighbors on the Web. Social Networks.

Li, Q.; Han, Z.; and Wu, X. M. 2018. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning.


Salha, G.; Vazirgiannis, M.; and Hennequin, R. 2020. Simple and Effective Graph Autoencoders with One-Hop Linear Models.


Sun, Z.; Deng, Z. H.; Nie, J. Y.; and Tang, J. 2019. RotateE: Knowledge Graph Embedding by Relational Rotation in Complex Space.


Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2017. Graph Attention Networks.

Wang, H.; Ren, H.; and Leskovec, J. 2020. Relational Message Passing for Knowledge Graph Completion.

Wang, X.; and Vinel, A. 2021. Benchmarking Graph Neural Networks on Link Prediction.


Weisfeiler, B. Y.; and Leman, A. A. 1968. A reduction of a Graph to a Canonical Form and an Algebra Arising during this Reduction (in Russian).


Zhang, J.; Zhang, H.; Xia, C.; and Sun, L. 2020a. Graph-Bert: Only Attention is Needed for Learning Graph Representations.


Zhang, M.; and Chen, Y. 2020. Inductive Matrix Completion Based on Graph Neural Networks. In International Conference on Learning Representations.


Zhou, T.; and Zhang, L. L.-C. 2009. Predicting missing links via local information. The European Physical Journal B.