Submission Template for the Cambridge University Press' Political Analysis Journal

Author One[1], Author Two[2], Author Three[3], and Author Four[2]

[1]*Department One, Institution One, Address One. Email: abc@example.edu*
[2]*Department Two, Institution Two, Address Two.*
[3]*Department Three, Institution Three, Address Three.*

1 Methodology

Girvan-Newman Algorithm

If you look at the structure of many graphs, you can see that some nodes are tightly connected, and some are loosely connected. And we can proceed with graph clustering using this. The Girvan-Newman algorithm uses betweenness centrality to measure the connectivity between each other and then finds the community of the network by finding the boundaries of the community. Betweenness centrality mainly expresses the relationship between nodes by using the short-path distance between nodes. The Girvan-Newman algorithm extends this to the case of an edge, measuring how important the edge is to connect nodes. Edge-betweenness measures how much the shortest distance between nodes has passed the corresponding edge. If the edge-betweenness of an edge is high, the edge is likely to be the main edge that connects communities.**girvan2002community**

The algorithm can detect communities hierarchically and is also used to detect communities in various fields such as social science, chemistry, and life science.

RandomWalk Positional Encoding

Many methods have been proposed to represent the graph structure. However, each of the previous methods has a problem. (add example) In the case of mainly used Laplacian PE, there is a problem that the sign values of the Laplacian eigenvector are unclear. (when selecting k eigenvector, 2^k sign values exist) To solve this problem, Dwivedi et al.(2022) **dwivedi2021graph** proposes Random Walk Positional Encoding.

Eq. 1

$$\begin{aligned} P_{ii} &= [RW_{ii}, RW_{ii}^2, RW_{ii}^3, ..., RW_{ii}^k], RW = AD^{-1} \\ P^{RWPE} &= [RW, RW^2, RW^3, ..., RW^k], RW = AD^{-1} \end{aligned} \tag{1}$$

The RWPE consists of a random walk value of k-steps based on the random walk diffusion process.(add formula) However, In the case of RWPE, there is a problem that isomorphic construction cannot be detected. Given enough steps, this can be solved, but in that case, the computation increases. To solve this problem, We inject additional features for the graph structure. Additional features used in this paper are a collection of transition matrices using adjacency matrix and hierarchical cluster embeddings obtained through the Girvan-Newman algorithm. They can all be expressed by the size of a tensor with dimension NxNxD. We also obtain Random Walk Positional Encoding through the successive matrix product of the random walk Laplacian matrix, which is also expressed through the inverse matrix of the degree matrix and the product of the adjacency matrix. Therefore, we used a technique of injecting in the middle of the transformer with additional features, rather than taking only the existing diagonal elements and adding them to the node data. The method may overcome a problem of a method of combining existing location information with node information.

Hierarchical Cluster Embedding by Dynamic Relu

Political Analysis (2023)

DOI: 10.1017/pan.xxxx.xx

Corresponding author
Author One

Edited by
John Doe

By applying the Girvan-newman Algorithm in the original graph, clustering is performed for each node. According to the hierarchical clustering of the algorithm, each node expresses that the node belongs to the cluster in a one-hot expression. Eq. 2

$$\theta^k(x) = Norm(f(Relu(f(AveragePooling(x)))))$$
$$\pi_{theta(x)}(x) = \max_{1 \le k \le K} (\theta^1(x)x, \theta^2(x)x, ...\theta^k(x)x, ..., \theta^K(x)x) \tag{2}$$

In addition, since the size of each cluster (the number of nodes belonging to the cluster) is different, Dynamic Relu was used to perform cluster embeddings that properly consider it. Dynamic Relu is an activation function that changes the slope according to the value of the corresponding dimension. First, average-pooling is performed for each channel, and the value is added to the hyper-function. At this time, since each cluster matrix is represented by one-hot vector, the value in hyper-function is the size of the corresponding cluster. The hyper-function dynamically allocates the slope according to the cluster size. Through this method, effective embedding is possible even with a small amount of computation. As a result, Hierarchical cluster data injection works effectively on datasets such as chemical molecules where hierarchical structures are strongly applied.

Figure 1

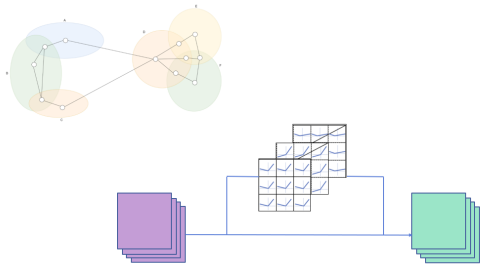
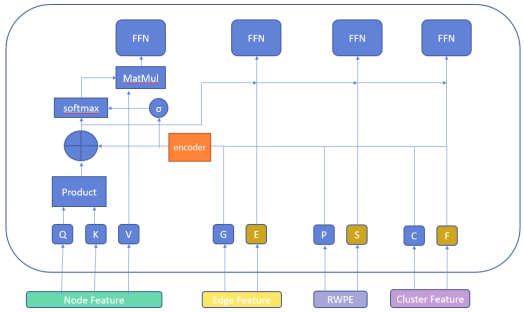


Figure 1. Herachical Cluster Embedding

Network Architecture



EGT**hussain2022global** is used as the backbone to consider the edges data. Similarly, Additional graph structure feature including positional encoding is also continuously injected and calculated through FFN layer by applying Learnable Structure, not adding node feature from the beginning. The additional features are combined in the encoder and injected into the transformer. The encoder sums each fit using a shared parameter. The structure of the encoder may cope with a situation in which additional features are further increased later. There is a problem in that information on the graph structure is lost as the layer increases in the existing graph neural network. We solve the problem by continuous injection of graph structure information through learnable structure using Feed Forward Network. In addition, the learnable parameter injects a suitable graph

structure into the corresponding node fit, enabling proper consideration of the graph structure.

$$
\begin{aligned}
Attention(Q, K, V) &= \acute{A}V \\
\acute{A} &= softmax(\frac{QK^T}{\sqrt{d_k}}) \\
H^q = HW^q, H^k &= HW^k, H^v = HW^v \\
E^{Edge} &= [adj, adj^2, adj^3, ..., adj^{hop}], G = \sigma(E) \\
E^{RWPE} &= [(D^{-1}A), (D^{-1}A)^2, (D^{-1}A)^3, ..., (D^{-1}A)^{hop}] \\
E^{RWPE} &= [cluster_1, cluster_2, ..]
\end{aligned}
$$

$$
Attention(H^q, H^k, H^v) = \acute{A}H^v \tag{3}
$$

$$
where, \acute{A} = softmax(\hat{H}) \odot \sigma(G) \tag{4}
$$

$$
where, \hat{H} = clip(\frac{H^q H^k}{\sqrt{d_k}}) + E \tag{5}
$$

$$
a_{ij} = \frac{1}{\sqrt{d}}(h_i^l W^{q,l})(h_j^l W^{k,l}) + e_{ij} \tag{6}
$$

$$
where, e_{ij} = f(e_{ij}^{Edge}) + f(e_{ij}^{RWPE}) + f(e_{ij}^{Hiera}) \tag{7}
$$

E and G are edge features. B is a learned scalar created using a short-path-distance matrix and enters the head collectively.

Node feature, Edge feature, and PE feature are calculated as described above term(4).**ke2020rethinking**

$$
\begin{aligned}
h_i^l &= h_i^{l-1} + O_h^l \parallel \sum_1^k \acute{A}V^l \\
e_{ij}^{l,Edge} &= e_i^{l-1} + O_e^l \parallel \sum_1^k \hat{H}_i^l \\
e_{ij}^{l,RWPE} &= e_i^{l-1} + O_e^l \parallel \sum_1^k \hat{H}_i^l \\
e_{ij}^{l,Hiera} &= e_i^{l-1} + O_e^l \parallel \sum_1^k \hat{H}_i^l
\end{aligned} \tag{8}
$$

O is learned output projection matrices. Node feature, edge feature, and positional encoding all perform multi-head attention operations. After that, the result of the attention each passes through the Feed Foward Network. Exceptionally, positional ecnoding concatenate with the feature of Node before passing FFN layer.

$$
\begin{aligned}
h_i^l &= h_i^{l-1} + FFN(LayerNorm(h_i)) \\
e_i^l &= e_i^{l-1} + FFN(LayerNorm(e_i))
\end{aligned} \tag{9}
$$

2 Results

-2*Model	ZINC MAE ↓		PATTERN %Accuracy ↑		CLUSTER %Accuracy ↑	MNIST %Accuracy ↑	CIFAR10 %Accuracy ↑
	#Param ≈ 100K	#Param ≈ 500K	#Param ≈ 100K	#Param ≈ 500K	#Param ≈ 500K	#Param ≈ 100K	#Param ≈ 100K
GCN	0.459 ± 0.006	0.367 ± 0.011	63.880 ± 0.074	71.892 ± 0.334	68.498 ± 0.976	90.705 ± 0.218	55.710 ± 0.381
GraphSage	0.468 ± 0.003	0.398 ± 0.002	50.516 ± 0.001	50.492 ± 0.001	63.844 ± 0.110	97.312 ± 0.097	65.767 ± 0.308
GIN	0.387 ± 0.015	0.526 ± 0.051	85.590 ± 0.011	85.387 ± 0.136	64.716 ± 1.553	96.485 ± 0.097	55.255 ± 1.527
GAT	0.475 ± 0.007	0.384 ± 0.007	75.824 ± 1.823	78.271 ± 0.186	70.587 ± 0.447	95.535 ± 0.205	64.223 ± 0.455
GT		0.226 ± 0.014		84.808 ± 0.068	73.169 ± 0.622		
GatedGCN	0.375 ± 0.003	0.214 ± 0.013	84.480 ± 0.122	86.508 ± 0.085	76.082 ± 0.196	97.340 ± 0.143	67.312 ± 0.311
PNA	0.188 ± 0.004	0.142 ± 0.010	86.567 ± 0.075			97.690 ± 0.022	70.350 ± 0.630
DGN	0.168 ± 0.003		86.680 ± 0.034				**72.700 ± 0.540**
Graphormer		0.122 ± 0.006		86.650 ± 0.033	74.660 ± 0.236	97.905 ± 0.176	65.978 ± 0.579
EGT	0.137 ± 0.006	**0.108 ± 0.005**	86.806 ± 0.032	86.837 ± 0.006	79.232 ± 0.348	**98.247 ± 0.058**	67.873 ± 1.02
Ours	**0.130 ± 0.001**	0.142 ± 0.015	**86.886 ± 0.038**	**86.977 ± 0.035**	**79.486 ± 0.151**	98.247 ± 0.087	64.23 ± 0.641

Table 1