[KDD 2020] GPT-GNN: Generative Pre-Training of Graph Neural Networks. [pdf] [code]

Node/Graph Tasks: node classification, rating scores classifications (edge weights prediction)

Training Type: pretraining and fine tuning

Pretext task data: structure and feature The pretext task here is to train a GNN encoder to extract node embeddings such that we can reconstruct the input graph's structure and attributes.

Initial short summary here This paper presents the GPT-GNN framework for generative pre-training of graph neural networks. This framework performs attribute and edge generation to enable the pre-trained model to capture the inherent dependency between node attributes and graph structure.

Formally, given an input graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ and a GNN model f_θ, the likelihood over this graph by this GNN model is $p(G;\theta)$ which represents how the nodes in G are attributed and connected. GPT-GNN aims to pre-train the GNN model by maximizing the graph likelihood, i.e., $\theta^* = \max_\theta p(G;\theta)$. Most existing graph generating methods follow the auto-regressive manner to factorize the probability objective such as the nodes in the graph coming in an order and the edges are generate by connecting each new arriving node to existing nodes. Given a purmutated order, we can factorize the log likelihood autoregressively - generating one node per iteration as:

$$\log p_\theta(X,E) = \sum_{i=1}^{|\mathcal{V}|} \log p_\theta(X_i, E_i | X_{<i}, E_{<i}) \tag{67}$$

At each step i, we use all nodes that are generated before i, their attributes $X_{<i}$, and the structure (edges) between these nodes $E_{<i}$ to generate a new node i, including both its attribute X_i and its connections with existing nodes E_i. Instead of directly assume that X_i, E_i are independent, i.e., $p_\theta(X_i, E_i | X_{<i}, E_{<i}) = p_\theta(X_i | X_{<i}, E_{<i}) \cdot p_\theta(E_i | X_{<i}, E_{<i})$, the paper devices the dependency-aware factorization mechanism to remain the dependency between node attributes and edge existence. The core idea is to when estimate a new nodes' attributes, we are given its structure information and vice versa. The generation process can be decomposed into two coupled parts: first generate node attributes given the observed edges and then generate the remaining edges given the observed edges and the generated node attributes. Denoting that $E_{i,o}$ as the observed edges within E_i while $E_{i,u}$ as the masked edges which are to be generated. THe conditional probability is rewritten as:

$$p_\theta(X_i, E_i | X_{<i}, E_{<i}) = \mathbb{E}_o(p_\theta(X_i | E_{i,o}, X_{<i}, E_{<i}) \cdot p_\theta(E_{i,u} | E_{i,o}, X_{\leq i}, E_{\leq i})) \tag{68}$$

Note that the observed edges are initially randomly selected to be remained in the graph. The attribute generation loss is defined as:

$$\mathcal{L}^{\text{Attr}} = \text{Distance}(\text{Dec}^{\text{Attr}}(h_i^{\text{Attr}}), X_i) \tag{69}$$

The edge reconstruction loss is defined as:

$$\mathcal{L}^{\text{Edge}} = -\sum_{j^+ \in E_{i,u}} \log \frac{\exp(\text{Dec}^{\text{Edge}}(h_i^{\text{Edge}}, h_{j^+}^{\text{Edge}}))}{\sum_{j \in S_i^- \cup \{j^+\}} \exp(\text{Dec}^{\text{Edge}}(h_i^{\text{Edge}}, h_j^{\text{Edge}}))} \tag{70}$$

where j^+ belongs the the edges that will be formed to connect with node i but has yet to be observed (positive samples) while S_i^- includes all edges that will not be connected with node i (negative samples).

To adapt the GPT-GNN to the heterogeneous graphs, the paper proposes to assign different decoders to differnet types of nodes and edges.

Bibtex:

@inproceedingshu2020gpt, title=Gpt-gnn: Generative pre-training of graph neural networks, author=Hu, Ziniu and Dong, Yuxiao and Wang, Kuansan and Chang, Kai-Wei and Sun, Yizhou, booktitle=Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages=1857–1867, year=2020