GPT-GNN: Generative Pre-Training of Graph Neural Networks

Kim Minhyoung

Index

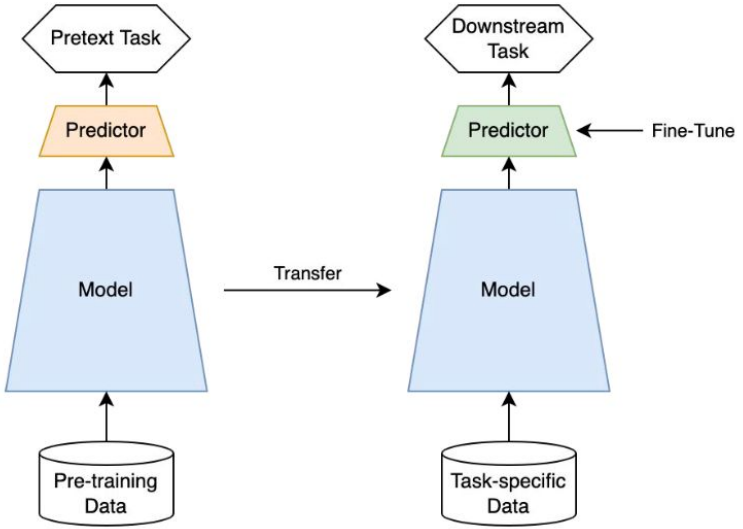
- Introduction
- Self-supervised learning
- GPT – Generative Pre-trained Transformer
- Graph Neural Network
- MessagePassing
- GAT – Graph Attention Network
- GPT-GNN
- Experiments

What is GPT-GNN framework?

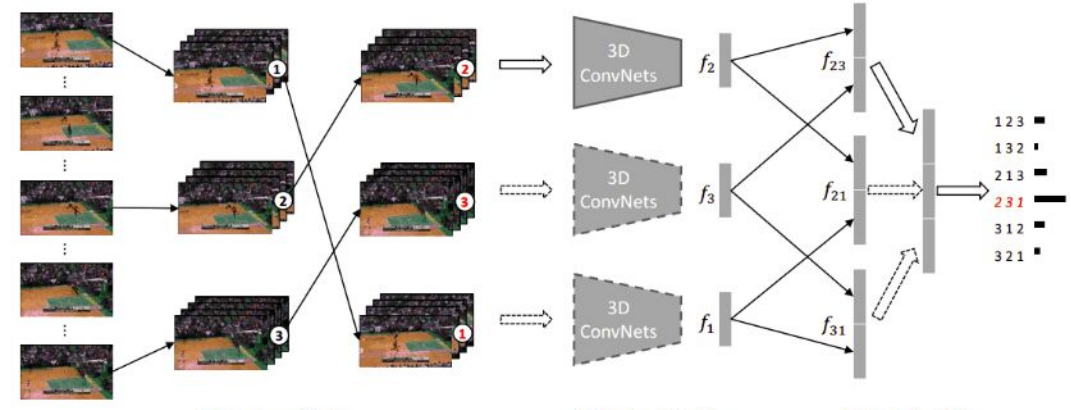
- Initialize Graph Neural Network by learning generative pretext task
- self-supervised learning
- Motivated from GPT (Generative Pre-trained Transformer)
- Graph Neural Network

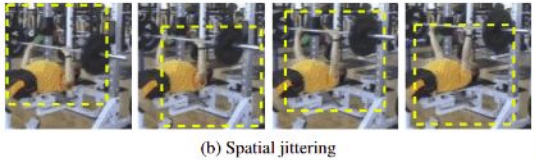
Self-supervised Learning

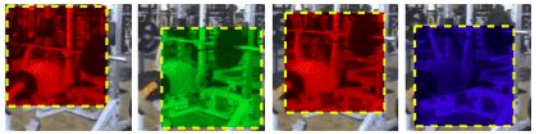
- Self-Supervised Learning은 unlabeled data에서 중요한 특징들을 추출해 내기 위하 Pretext Task를 설계하여 Unsupervised Learning을 진행하여 다양한 Task에서 중요한 정보를 추출할 수 있게 한다
- 이 모델을 Downstream Task에 대해 Transfer Learning을 진행하여 해당 테스크에 더 높은 정확도를 가지게 할 수 있다.

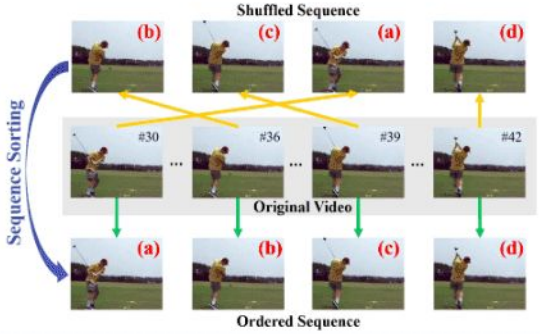


Self-supervised Learning



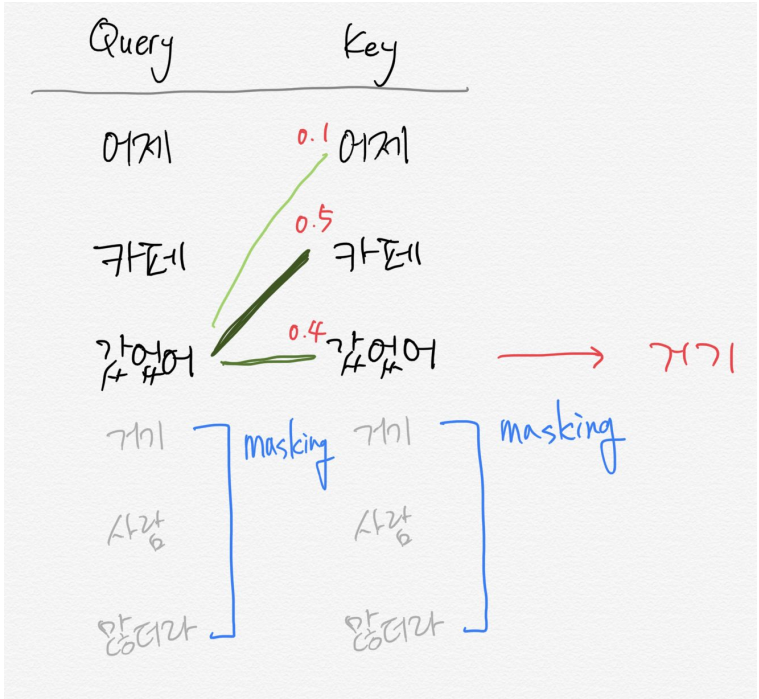

(b) Spatial jittering


(c) Channel splitting



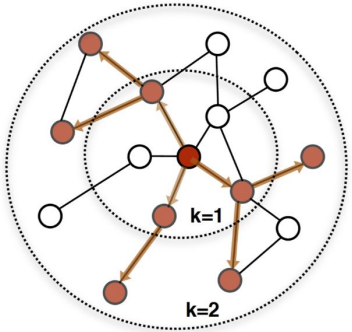
GPT – Generative Pre-trained Transformer

- GPT도 대표적인 Self-Supervised Learning
- 이전 단어들을 주고 다음 단어를 맞추는 것을 학습
- Attention의 구조는 전체 문장의 정보를 볼 수 있으므로 Attention 값에 0을 곱하여 모델이 정보를 얻지 못하게 하는 maskng 사용

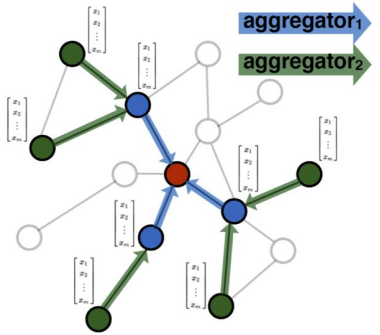


Graph Neural Network

- Deep Learning의 기존 연구는 정형화된 데이터, 이미지, 음성 등 Euclidean Data를 가지고 진행되었음
- Non Euclidean Data인 그래프 데이터를 가지고 모델을 구축하기 어려움
- 명확한 순서를 가지고 있는 Euclidean Data와 다르게 Non Euclidean Data는 정해진 순서가 없음
- 이런 데이터에서 Neural Network를 활용하기 위한 시도로 GraphSAGE 같은 알고리즘이 제안되었으며, 최근 활발하게 연구되는 분야임



1. Sample neighborhood



2. Aggregate feature information from neighbors

MessagePassing

- GraphSAGE에서 제안된 일반화된 GNN 설계 구조가 MessagePassing
- 이웃들로 부터 Message (feature)를 전달 받아 Aggregate하여 새로운 feature로 출력함
- Aggregate된 Message에 노드 자신의 feature도 반영한 뒤 모델의 출력으로 생성함

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

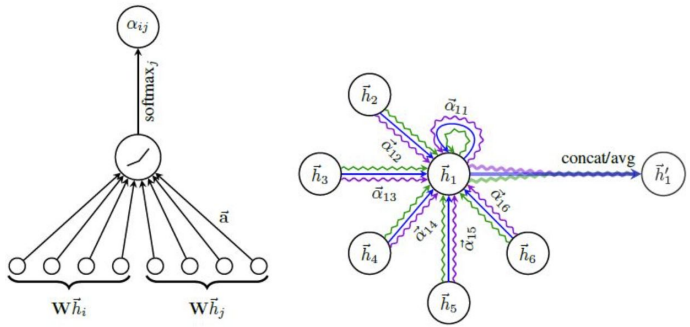
$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

GAT – Graph Attention Network

- GAT는 MessagePassing 구조를 따르면서 Attention을 적용한 네트워크
- parameter vector와 가중치행렬 W를 학습
- 이웃들의 feature에 normalized attention score를 곱하여 합산한 뒤 비선형성 추가한 것을 출력함

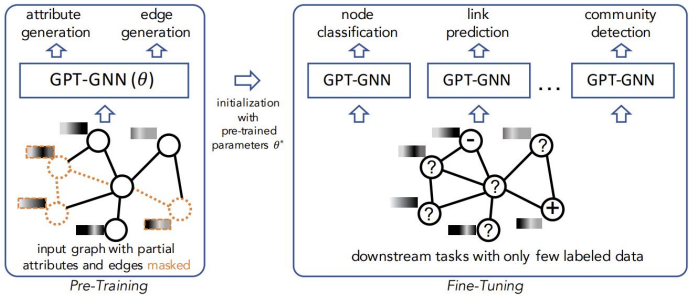
$$\alpha_{ij} = \frac{exp(LeakyRELU(\vec{a}^T[\mathbf{W}\vec{h}_i|\mathbf{W}\vec{h}_j]))}{\Sigma_{k\in N_i} exp(LeakyRELU(\vec{a}^T[\mathbf{W}\vec{h}_i|\mathbf{W}\vec{h}_k]))}$$

$$\vec{h}_i' = \sigma(\Sigma_{j\in N_i}\alpha_{ij}\mathbf{W}\vec{h}_j)$$



GPT-GNN

- GPT-GNN은 그래프의 permutation을 정했을때 노드가 추가되면서 기존에 있는 노드와 연결되는 edge가 추가되는 구조에서 착안
- 노드 속성이 있는 그래프 G에서 feature 정보와 structural 정보를 같이 학습함
- 그래프를 재생성 할때 기존 그래프의 정보를 바탕으로
- 새 노드를 추가하고 그 노드의 특징을 gnn의 representation으로 만들며, 그 노드와 연결되는(연결되지 않는) 노드를 예측



GPT-GNN

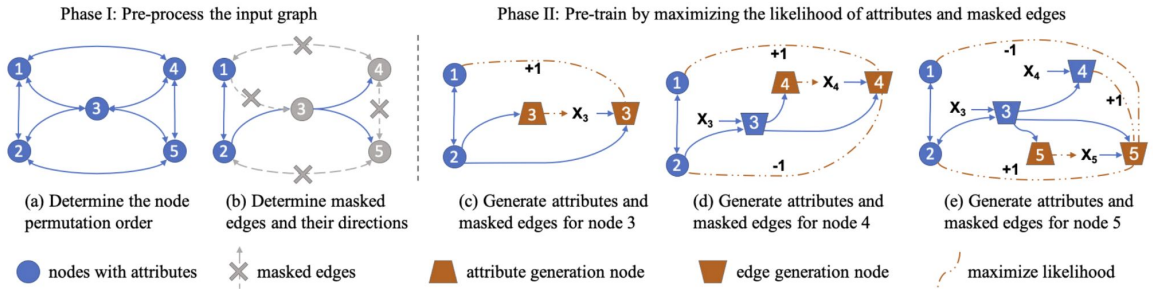
- 이때 생성하는 노드의 feature와 edge 예측이 독립적이면 모델 성능이 저하됨
- 실제 그래프는 속성과 edge가 관련있음
- 따라서 이를 두 단계로 분해하여 학습
- 노드의 feature는 edge와 독립적으로 생성함
- edge prediction은 생성한 노드 feature의 정보를 포함하여 edge connecting representation을 생성

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

$$\Rightarrow \quad \mathbb{E}_o\Big[\underbrace{p_\theta(X_i \mid E_{i,o}, X_{<i}, E_{<i})}_{\text{1) generate attributes}} \cdot \underbrace{p_\theta(E_{i,\neg o} \mid E_{i,o}, X_{\leq i}, E_{<i})}_{\text{2) generate edges}}\Big]$$

GPT-GNN

- 전체 그래프에 대해 학습을 진행하면 dependency가 강함
- 그래프 샘플링을 통해 학습
- 학습에서 정보 손실을 막기 위해 노드를 두 가지로 분리해서 학습함
- Attribution Generation Node – feature 생성 노드, mask 토큰과 동일한 효과를 주는 벡터 X_init을 masking 되는 노드의 feature로 입력
- Edge Generation Node – edge 생성은 서로 독립적인 과정으로 봐서 masking X
- 각 노드가 생성한 repr에 적절한 decoder를 붙여 학습



Loss

- Attribute Decoder는 feature type에 따라 MLP, LSTM 등을 사용 가능
- Distance Function (L2-Norm 등등)을 통해 Loss 정의
- Edge Decoder는 pairwise score (두 노드가 연결될지 되지 않을지) 사용
- 이웃이 없는 노드는 학습할때 더 많이 학습되도록 계산시 포함함

$$\mathcal{L}_i^{Attr} = Distance\left(Dec^{Attr}(h_i^{Attr}), X_i\right)$$

$$\mathcal{L}_i^{Edge} = -\sum_{j^+ \in E_{i,\neg o}} \log \frac{\exp\left(Dec^{Edge}(h_i^{Edge}, h_{j^+}^{Edge})\right)}{\sum_{j \in S_i^- \cup \{j^+\}} \exp\left(Dec^{Edge}(h_i^{Edge}, h_j^{Edge})\right)}$$

Experiments

- Open Academic Graph (OAG)
- Amazon Recommendation Dataset
- OAG Task – Prediction of Field, Venue, Author ND
- Amazon Task – Prediction Score of Review (Fashion, Beauty, Luxury fields)

Experiments

	Downstream Dataset	OAG			Amazon		
	Evaluation Task	Paper–Field	Paper–Venue	Author ND	Fashion	Beauty	Luxury
	No Pre-train	.336±.149	.365±.122	.794±.105	.586±.074	.546±.071	.494±.067
Field Transfer	GAE	.403±.114	.418±.093	.816±.084	.610±.070	.568±.066	.516±.071
	GraphSAGE (unsp.)	.368±.125	.401±.096	.803±.092	.597±.065	.554±.061	.509±.052
	Graph Infomax	.387±.112	.404±.097	.810±.084	.604±.063	.561±.063	.506±.074
	GPT-GNN (Attr)	.396±.118	.423±.105	.818±.086	.621±.053	.576±.056	.528±.061
	GPT-GNN (Edge)	.401±.109	.428±.096	.826±.093	.616±.060	.570±.059	.520±.047
	GPT-GNN	**.407±.107**	**.432±.098**	**.831±.102**	**.625±.055**	**.577±.054**	**.531±.043**
Time Transfer	GAE	.384±.117	.412±.101	.812±.095	.603±.065	.562±.063	.510±.071
	GraphSAGE (unsp.)	.352±.121	.394±.105	.799±.093	.594±.067	.553±.069	.501±.064
	Graph Infomax	.369±.116	.398±.102	.805±.089	.599±.063	.558±.060	.503±.063
	GPT-GNN (Attr)	.382±.114	.414±.098	.811±.089	.614±.057	**.573±.053**	.522±.051
	GPT-GNN (Edge)	.392±.105	.421±.102	.821±.088	.608±.055	.567±.038	.513±.058
	GPT-GNN	**.400±.108**	**.429±.101**	**.825±.093**	**.617±.059**	.572±.059	**.525±.057**
Time + Field Transfer	GAE	.371±.124	.403±.108	.806±.102	.596±.065	.554±.063	.505±.061
	GraphSAGE (unsp.)	.349±.130	.393±.118	.797±.097	.589±.071	.545±.068	.498±.064
	Graph Infomax	.360±.121	.391±.102	.800±.093	.591±.068	.550±.058	.501±.063
	GPT-GNN (Attr)	.364±.115	.409±.103	.809±.094	.608±.062	.569±.057	.517±.057
	— (w/o node separation)	.347±.128	.391±.102	.791±.108	.585±.068	.546±.062	.497±.062
	GPT-GNN (Edge)	.386±.116	.414±.104	.815±.105	.604±.058	.565±.057	.514±.047
	— (w/o adaptive queue)	.376±.121	.410±.115	.808±.104	.599±.068	.562±.065	.509±.062
	GPT-GNN	**.393±.112**	**.420±.108**	**.818±.102**	**.610±.054**	**.572±.063**	**.521±.049**

Table 1: Performance of different downstream tasks on OAG and Amazon by using different pre-training frameworks with the heterogeneous graph transformer (HGT) [15] as the base model. 10% of labeled data is used for fine-tuning.

감사합니다