Graph-Bert：没有我Attention解决不了的

原创 kaiyuan NewBeeNLP 2020-08-17

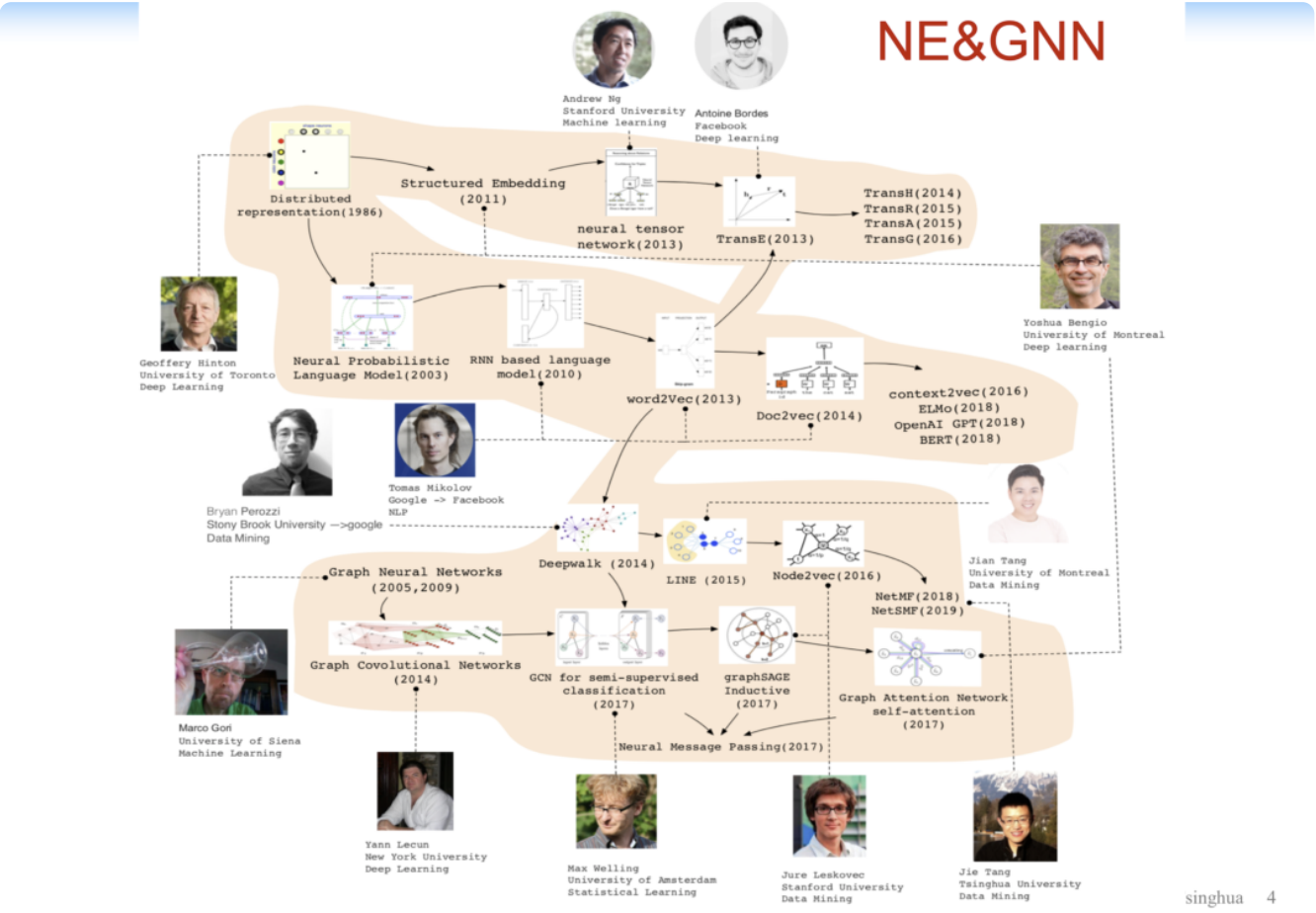
收录于话题
#图网络学习

9个

听说星标这个公众号⏫
模型效果越来越好噢 🥰

作者 | kaiyuan
整理 | NewBeeNLP

最近要做一个图相关的项目，之前没怎么接触过图网络，就从头开始学习了一下，后面有时间也会整理分享。这里顺便推荐一下清华大学唐杰老师的一个分享：**「图表示学习和图神经网络的最新理论进展」**，主要介绍了图神经网络及其在认知推理方向的一些进展。



ok，今天这篇文章主要是记录下**Graph-Bert**的阅读笔记，跟我们现在要做的比较像，是关于**「图网络的预训练」**。这一块还有很多非常棒的资料，后续再慢慢整理分享吧。

- 论文：https://arxiv.org/abs/2001.05140
- 代码：https://github.com/jwzhanggy/Graph-Bert

enjoy~

TL;DR

在这篇文章中，作者指出了目前主流GNN完全依赖图链接，存在一些严重影响性能的问题

- **模型假死(suspended animation)**：随着训练层数加深，模型不再对输入数据响应；
- **过度平滑(over-smoothing)**：随着训练层数加深，所有结点的embedding会越来越相似；
- **并行化处理(paralelization)**：在大图上无法并行化处理

针对以上问题，提出了Graph-BERT，同样是非常火的『pretrain+fintune』范式应用于图网络中。中心思想还是attention，首先将原始大图采样为一个个的子图，然后利用attention去学习结点表征。下面来看看具体模型👇

Model

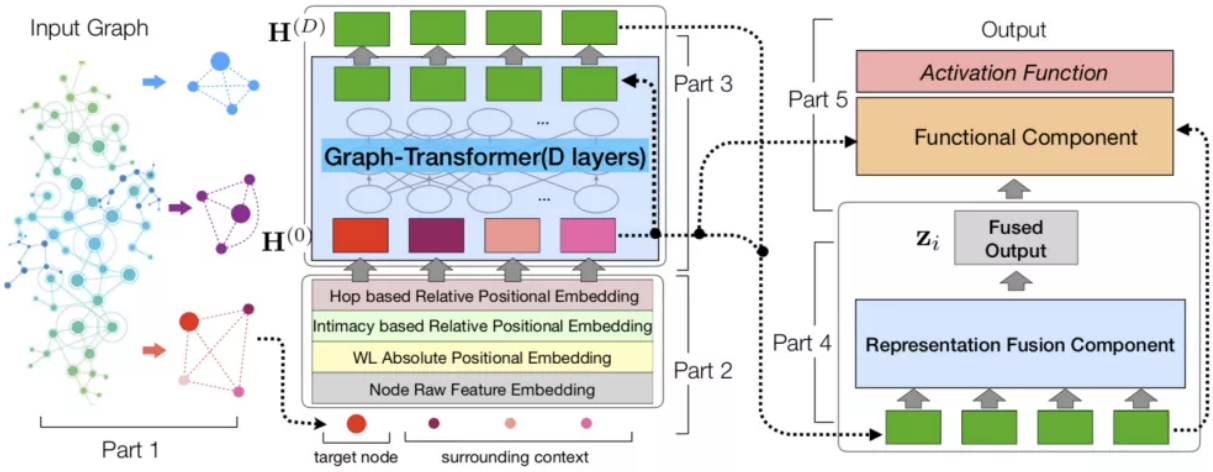


Figure 1: Architecture of the GRAPH-BERT Model. (Part 1: linkless subgraph batching; Part 2: node input vector embeddings; Part 3: graph transformer based encoder; Part 4: representation fusion; Part 5: functional component. Depending on the target application task, the function component will generate different output. In the sampled subgraphs, it covers both the target node and the surrounding context nodes.)

如上图，Graph-Bert模型主要可以分为四部分：

- 从原始大图中采样无连接子图（linkless graph）
- 输入结点embedding处理

- 基于图的transformer-encoder
- 基于图的transformer-decoder

子图采样

为了更好地处理大图（并行化），graph-bert选择在采样子图上进行训练。使用了一种基于图亲密度矩阵 $\mathbf{S} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ 的采样方案：**「top-k intimacy」**，其中 $S(i,j)$ 表示节点 v_i 和 结点 v_j 之间的亲密度，计算公式为：

$$\mathbf{S} = \alpha \cdot (\mathbf{I} - (1-\alpha) \cdot \overline{\mathbf{A}})^{-1}$$

其中，α 是一个[0，1]之间的超参数（通常取0.15），$\overline{\mathbf{A}} = \mathbf{A}\mathbf{D}^{-1}$ 表示列规范化邻接矩阵，A 为图的邻接矩阵， D 为对应的对角矩阵 $\mathbf{D}(i,i) = \sum_j \mathbf{A}(i,j)$。

那么对于一个给定的目标结点，就可以利用上面定义的亲密度来找出其上下文结点，计算公式为：

$$\mathbf{\Gamma}(v_i) = \{v_j \mid v_j \in \mathcal{V} \setminus \{v_i\} \wedge \mathbf{S}(i,j) \geq \theta_i\}$$

其实这一步就是把图结构的数据转变成了我们平时常见的NLP序列化输入，把每个结点看成是一个个字或词，然后后面就可以直接套transformer了。记得之前有篇文章说的也是类似的：Transformers are Graph Neural Networks[1]

结点embedding

由于经过采样出来的结点们是无序的，这里按照**「与target node的亲密度打分」**来对结点集合进行排序。结点emdedding由四部分组成

「1. 原始特征embedding」

就是使用一个映射操作将原始特征表示到新的共享的特征空间，对于不同的输入可以有不同的映射函数，如CNN/LSTM/BERT等

$$\mathbf{e}_j^{(x)} = \mathrm{Embed}(\mathbf{x}_j) \in \mathbb{R}^{d_h \times 1}$$

「2. Weisfeiler-Lehman 绝对角色embedding」

Weisfeiler-Lehman算法是用来确定两个图是否是同构的，其基本思路是通过迭代式地聚合邻居节点的信息来判断当前中心节点的独立性(Identity)，从而更新整张图的编码表示。

$$\begin{aligned}\mathbf{e}_j^{(r)} &= \text{Position-Embed}\left(\mathbf{WL}(v_j)\right) \\ &= \left[\sin\left(\frac{\mathbf{WL}(v_j)}{10000^{\frac{2l}{d_h}}}\right), \cos\left(\frac{\mathbf{WL}(v_j)}{10000^{\frac{2l+1}{d_h}}}\right)\right]_{l=0}^{\left\lfloor\frac{d_h}{2}\right\rfloor}\end{aligned}$$

有关更多WL算法的细节可以参考这个slides：Graph Kernel[2]

「3．基于亲密度的相对位置embedding」

上一节计算的嵌入可以表示全局的信息，而这一步主要是获取局部信息。

$$\mathbf{e}_j^{(p)} = \text{Position-Embed}\left(\mathrm{P}(v_j)\right) \in \mathbb{R}^{d_h \times 1}$$

「4．基于相对距离embedding」

对两个结点在原始大图中的距离（间隔边的数量）进行embedding表示，主要是为了平衡上述两步的embedding

$$\mathbf{e}_j^{(d)} = \text{Position-Embed}\left(\mathrm{H}(v_j; v_i)\right) \in \mathbb{R}^{d_h \times 1}$$

Transfomer编码器

「输入」

首先是对前面得到的四个embeeding进行聚合，

$$\mathbf{h}_j^{(0)} = \text{Aggregate}\left(\mathbf{e}_j^{(x)}, \mathbf{e}_j^{(r)}, \mathbf{e}_j^{(p)}, \mathbf{e}_j^{(d)}\right)$$

聚合的函数有很多可以选择，文章里作者就使用了最简单的加和操作。聚合之后就可以得到所有结点的输入表示：$H^{(0)} = \left[\mathbf{h}_i^{(0)}, \mathbf{h}_{i1}^{(0)}, \cdots, \mathbf{h}_{ik}^{(0)}\right]^\top \in \mathbb{R}^{(k+1)\times d_h}$

「更新」

然后就是进行N层的transformer encoder的迭代更新，

$$\begin{aligned}\mathbf{H}^{(l)} &= \text{G-Transformer}\left(\mathbf{H}^{(l-1)}\right) \\ &= \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_h}}\right)\mathbf{V} + \mathbf{G} - \text{Res}\left(\mathbf{H}^{(l-1)}, \mathbf{X}_i\right)\end{aligned}$$

「输出」

经过D层的编码之后，我们就可以得到对应每个结点的表示，$H^{(D)}=\left[\mathbf{h}_i^{(D)},\mathbf{h}_{i1}^{(D)},\cdots,\mathbf{h}_{ik}^{(D)}\right]^\top$，之后就可以根据具体的下游任务来使用这些向量表示。

预训练

图神经网络的预训练大致分为两部分：对结点的预训练以及对链接关系的预训练。

结点预训练

对于目标结点 v_i ，原始特征为 x_i，我们通过GRAPH-BERT编码层可以得到其隐藏表示 z_i，然后经过一层FC映射后 $\hat{\mathbf{x}}_i=\mathrm{FC}(\mathbf{z}_i)$ 重建原始特征。

$$\ell_1=\frac{1}{|\mathcal{V}|}\sum_{v_i\in\mathcal{V}}\|\mathbf{x}_i-\hat{\mathbf{x}}_i\|_2$$

链接关系预训练

这一部分其实是考虑了图的结构信息，利用两个节点表示的cos距离与之前提前算好的亲密度打分，来做损失。

$$\ell_2=\frac{1}{|\mathcal{V}|^2}\|\mathbf{S}-\hat{\mathbf{S}}\|_F^2$$

其中，$\hat{\mathbf{S}}(i,j)=\hat{s}_{i,j}$

微调

在文章中介绍了两个微调任务：**「节点分类」**和**「图聚类」**，效果看着都不错的样子。最后作者还加上了ablation study，分析了一些模型设置的效果，比如采样子图的结点数k，初始embedding，是否进行预训练等等。

本文参考资料

[1] **Transformers are Graph Neural Networks:** https://graphdeeplearning.github.io/post/transformers-are-gnns/

[2] **Graph Kernel:** https://www.slideshare.net/pratikshukla11/graph-kernelpdf