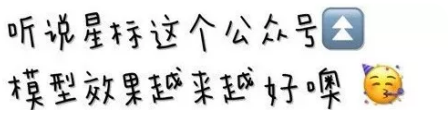
LINE：不得不看的大规模信息网络嵌入

原创 kaiyuan NewBeeNLP 2020-11-03

收录于话题
#图网络学习

9个



> NewBeeNLP原创出品
> 作者 | kaiyuan

和DeepWalk一样，今天介绍的论文同样是做网络嵌入表示的，但还是有很大区别的。关于DeepWalk，我们已经在之前文章介绍，戳：**DeepWalk：图网络与NLP的巧妙融合**

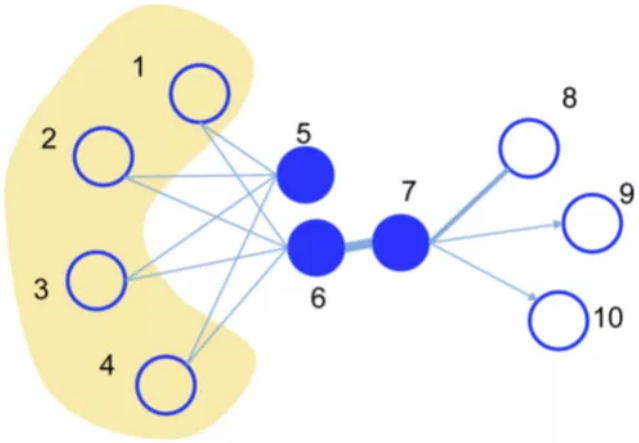
论文来自2015年微软，

- 论文：LINE: Large-scale Information Network Embedding
- 链接：https://arxiv.org/abs/1503.03578
- 源码：https://github.com/tangjianpku/LINE

从论文标题就可以看出，文章主打大规模图网络。当时大多数的嵌入表示研究在小型图网络上表现非常不错，但是当网络规模扩展到**百万、百亿级**别时，就会显得不尽人意。此外，适用场景也比较有限，无法应用到有向或者带权重图中。为此，本文提出了一种新的网络向量嵌入模型LINE，以解决上述等问题。

先验知识

在我们介绍模型之前，以如下示例先来了解一下相关概念定义。



一阶相似性

一阶相似性定义为两个顶点 u 和 v 之间的邻近度，用该边的权重 W_{uv} 表示，如果两个顶点之间没有边，那么它们的一阶相似性为0。这个概念是用于模型刻画局部信息的。

如上图，一阶相似性的大小就可以用链接线的粗细来表示。

二阶相似性

在真实场景中，大规模图中有链接的结点相对少，因此如果只用上述一阶相似性来建模是不全面的。比如上图中的5和6结点，两者没有链接，但是拥有几乎完全相同的邻居结点，我们可以认为它们的距离应该也是近的。

二阶相似性定义为一对结点的邻居网络结构相似性。类比到NLP中就是上下文的相似性，即经典的『**you shall know a word by the company it keeps**』。这个概念用于模型刻画全局信息。

KL散度

KL散度是用于衡量两个概率分布相似性的指标，定义为：

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) \cdot (\log p(x_i) - \log(q(x_i))$$

表示概率分布 p 和概率分布 q 之间的差异，越小越接近。

LINE模型

一阶相似性的LINE模型

对于两个顶点 v_i 和 v_j ，它们之间的相似性可以用向量距离来表示(其中 u_i 和 u_j 分别表示对应的向量)

$$p_1(v_i, v_j) = \frac{1}{1 + \exp\left(-\vec{u}_i^T \cdot \vec{u}_j\right)}$$

而实际直观上两个结点的相似度是用链接强度表示，即边的权重，可以表示为，

$$\hat{p}_1(i, j) = \frac{w_{ij}}{W}$$

因此我们的目标函数就是使得 $p_1(v_i, v_j)$ 和 $\hat{p}_1(i, j)$ 尽可能地相同。论文里使用了上一节介绍的KL散度，

$$O_1 = -\sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j)$$

注意，一阶相似度仅适用于无向图。

二阶相似性的LINE模型

二阶相似性模型和word2vec类似，认为中间结点的上下文结点交集越大则越相似。对于每个节点 v_i 都有两个向量表示：一个是作为中间结点时的表示 $\vec{u_i}$，以及作为上下文结点时的表示 $\vec{u_i}'$ 。对于每一条边 (i, j)， 由结点 v_i 生成上下文 v_j 的概率为：

$$p_2(v_j \mid v_i) = \frac{\exp\left(\vec{u}_j^T \cdot \vec{u}_i\right)}{\sum_{k=1}^{|V|} \exp\left(\vec{u}_k'^T \cdot \vec{u}_i\right)}$$

啊哈！这不就是word2vec计算词向量的公式嘛！

实际直观上两个结点的二阶相似性可以表示为，

$$\hat{p}_2(v_j \mid v_i) = \frac{w_{ij}}{d_i}$$

其中 w_{ij} 为边的权重， d_i为结点的出度。最终也是通过KL散度来最小化两个概率分布的差距，

$$O_2 = -\sum_{(i,j)\in E} w_{ij} \log p_2(v_j \mid v_i)$$

注意，二阶相似度既可用于无向图，也可用于有向图、带权图。

模型优化

二阶相似性的目标函数中，$p_2(v_j \mid v_i)$ 这一项的计算会涉及所有和结点 i 相邻结点的内积，计算量很大。为此作者采用了『负采样』的方式进行优化，其中第一项为正样本的边，第二项为采样的负样本边。

$$\log \sigma\left(\vec{u}_j^T \cdot \vec{u}_i\right)+\sum_{i=1}^K E_{v_n \sim P_n(v)}\left[\log \sigma\left(-\vec{u}_n'^T \cdot \vec{u}_i\right)\right]$$

然后，当模型在优化更新过程中，对结点embedding的计算如下，

$$\frac{\partial O_2}{\partial \vec{u}_i}=w_{ij} \cdot \frac{\partial \log p_2(v_j \mid v_i)}{\partial \vec{u}_i}$$

很明显，当边的权重存在较大的方差时，会导致学习不稳定，无法选择一个合适的学习率。不难想到如果边的权重都相同，这个问题不就解决了。于是一个简单的做法是将权重为 w 的边拆分成 w 条binary edge，但是如果w很大则会很费存储空间。

一种更合理的思路是对边进行采样，采样概率正比于边的权重，然后把被采样到的边认为是binary edge处理。

思考

实验部分就略过了，感兴趣的小伙伴可以自行研究~

针对实际应用，作者提出了两个思考。

『孤岛』结点

原文中为『low degree vertices』，指的是拥有较少邻居结点的结点。这样一来，就很难学习到其向量表示，作者的建议是可以考虑邻居的邻居结点，即多采样几跳结点，丰富信息表示。

『新来』结点

对于一个新来的结点，如果它和已知的结点有链接关系，那么可以来优化下面任意一个目标函数，

$$-\sum_{j \in N(i)} w_{ji} \log p_1(v_j, v_i)$$
$$-\sum_{j \in N(i)} w_{ji} \log p_2(v_j \mid v_i)$$

并且保持原有已知节点的embedding不变，更新新来节点的embdding。

一起交流

重磅推荐！NewBeeNLP目前已经建立了多个不同方向交流群（**机器学习 / 深度学习 / 自然语言处理 / 面试交流 / 推荐系统 / 大厂内推** 等），赶紧添加下方微信加入一起讨论学习吧！

- END -

往期推荐

DeepWalk：图网络与NLP的巧妙融合

2020-09-21

Graph-Bert：没有我Attention解决不了的

2020-08-17

Transformers Assemble（PART V）

2020-03-10

ICLR2020 | 深度自适应Transformer

2020-07-20

Transformer温故知新