DeepWalk: 图神经网络节点嵌入

原创 韩锋 AI LAB plus 2020-08-15

DeepWalk: Online Learning of Social Representations

论文题目:

DeepWalk: Online Learning of Social Representations

论文链接:

http://www.perozzi.net/publications/14_kdd_deepwalk.pdf

代码连接:

https://github.com/phanein/deepwalk

轻音乐喜欢可以边听边阅读

A Little Story

Valentin - My View

引言

文章提出了一种学习图中节点嵌入的潜在表示的方法，这些潜在的表示法在连续向量空间中进行编码，很容易被统计模型利用。DeepWalk是一种在线、可扩展、并行的学习方法。很适合应用到节点分类、异常检测等场景。

幂次定律

"幂次定律"也叫“80-20法则”，由经济学家维尔弗雷多.帕累托在1906年提出，他认为：在任何一组东西中，最重要的只占其中一小部分，约20%，其余80%尽管是多数，却是次要的。

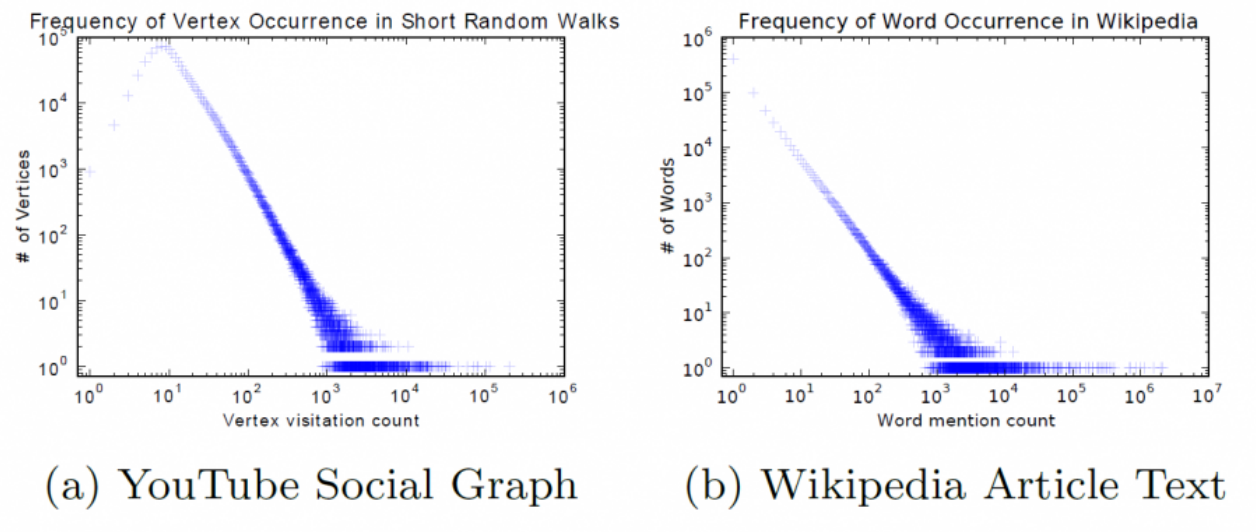


图1 幂次定律对比

自然语言处理中的单词频率已被证明遵循幂次法则，来自语言模型的技术解释了这种分布现象。所以，只需要证明，图中的数据也遵循幂次定律，那么就自然可以应用nlp中的方法来表示。如图，比较了对图进行短随机游走中向量出现的频率与单词在文本信息中出现的频率，发现图的短随机游走也大致满足幂次定律。

文章运用nlp中word2vec应用到图的表示上，把图中对节点的一串随机游走序列对应于word2vec中的上下文信息，表示成向量，进而为后续算法做准备。

输入与输出

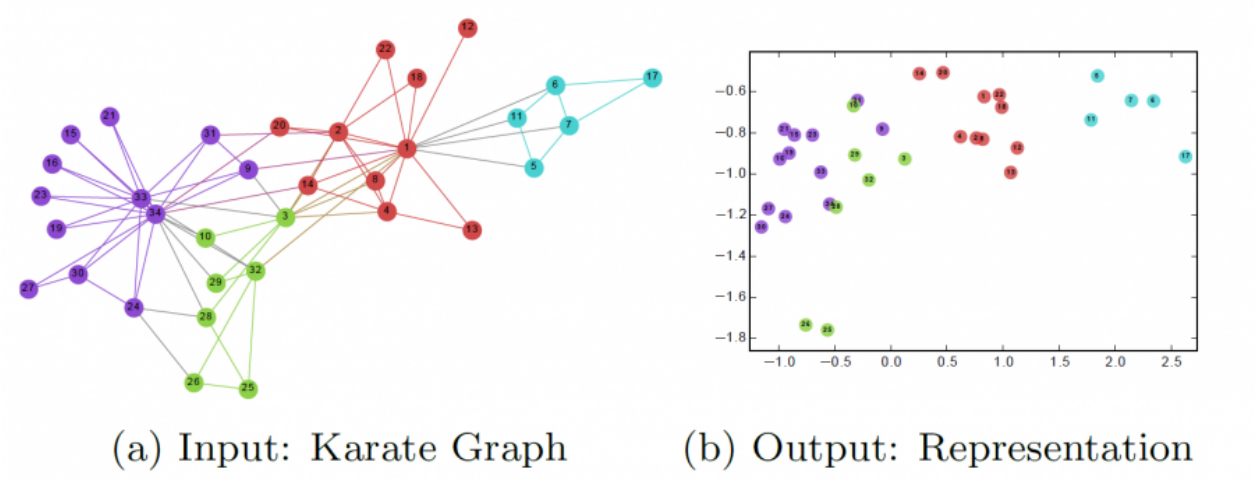
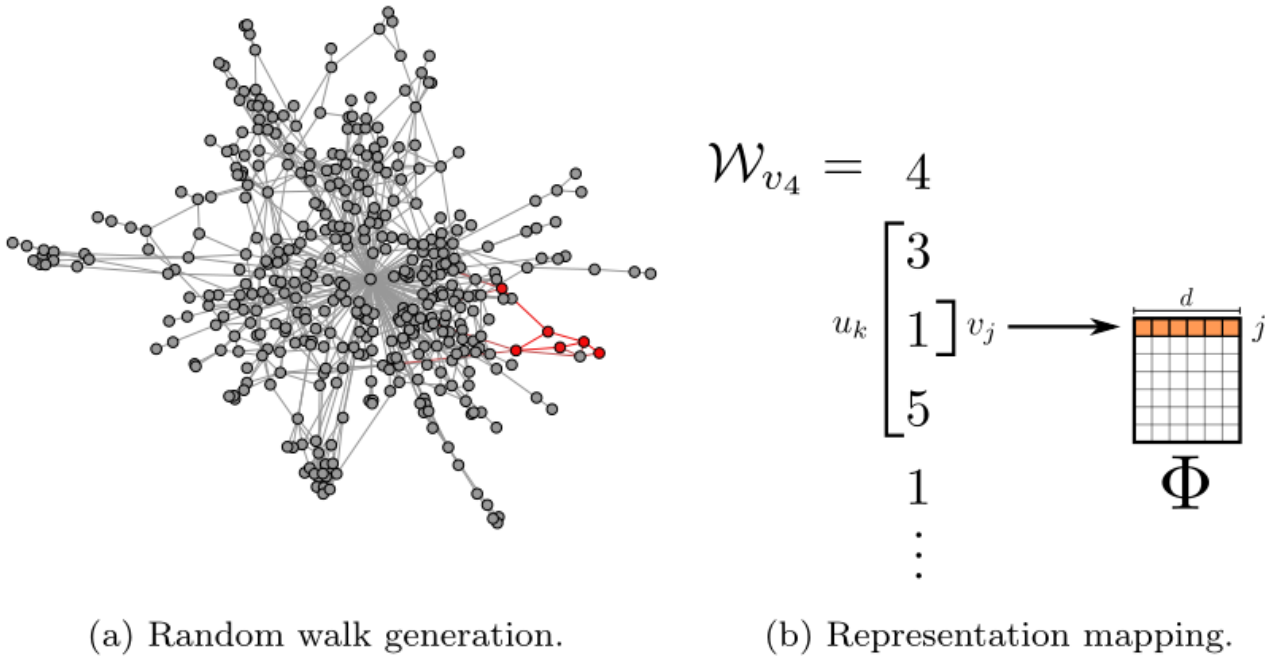


图2 输入与输出

输入是一个graph的点集和边集，输出是一个二维（x表示特征，y为标签的集合）表示，文章的目的就是学习得到x的低维表示。我们可以看到，具有相同标签的节点被聚集到一起，而不同标签的节点也大多数被分开了。

主要步骤



(a) Random walk generation. (b) Representation mapping.

图3 Random Walk and Skip-gram

Random Walk

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
  $\mathcal{O} = \text{Shuffle}(V)$
  for each $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, t)$
    SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
  end for
end for

图4 Random Walk 算法

如图是算法的整个流程，3-9行为核心。外循环制定了随机游走的次数，即将每次迭代都看作一次“传递”，并在每次传递过程中遍历一次节点。在开始遍历前，都会随机生成一个遍历顺序。在内循环中，遍历图上所有节点，生成一个节点序列，通过skip-gram学习节点的嵌入。

Skip-gram

Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

for each $v_j \in \mathcal{W}_{v_i}$ **do**
  for each $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ **do**
    $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
    $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
  end for
end for

图5 Skip-gram算法

skip-gram是word2vec中的一种语言模型，定义一个滑动窗口，skip-gram会最大化单词同时出现在这个窗口的概率。

Hierarchical Softmax

正如word2vec中，vocabulary的大小决定了Skip-Gram神经网络将会拥有大规模的权重矩阵，所有的这些权重需要通过数以亿计的训练样本来进行调整，这是非常消耗计算资源的，并且实际中训练起来会非常慢。本文也面临同样的问题，文章采用分层softmax来解决由于节点数量庞大而导致的 softmax 计算成本高昂的问题。

Hierarchical Softmax概述

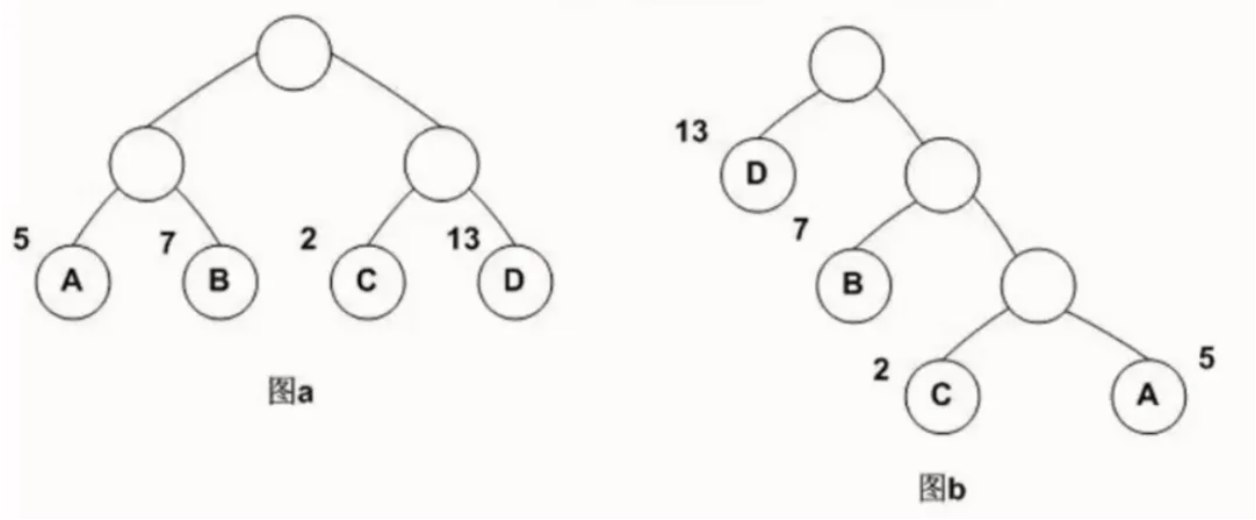


图6 树结构

图中数字表示权重，表示节点重要程度图a是常见的二叉树，图b就是图a转换过的最优二叉树。可以看出，D是最重要的，一个原则就是最重要的节点放在最前面，由此构造了图b的哈夫曼树。

它们的带权路径长度分别为：

图a：WPL = 5 * 2 + 7 * 2 + 2 * 2 +13 * 2 = 54

图b：WPL = 5 * 3 + 2 * 3 + 7 * 2 + 13 * 1 = 48

可见，图b哈夫曼树的带权路径长度较小。

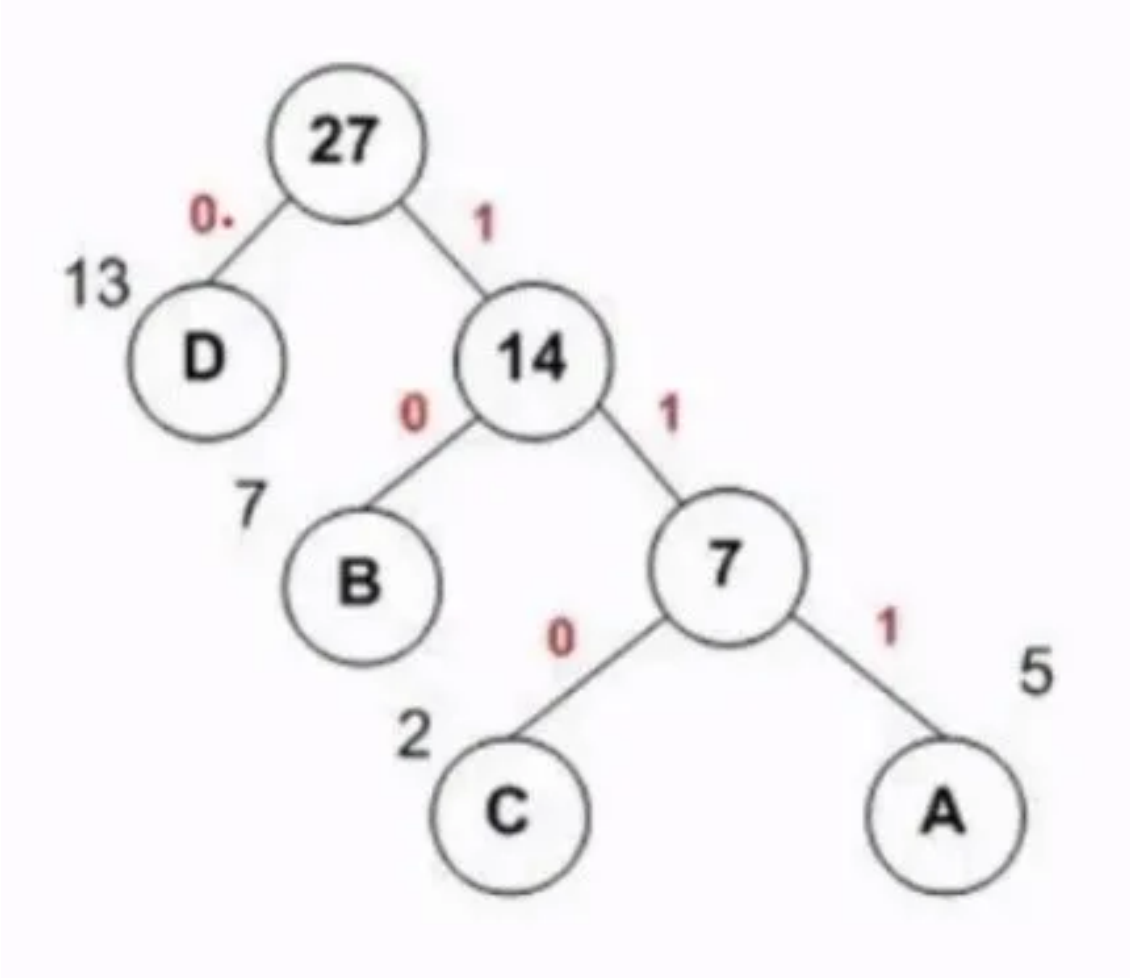


图7 哈夫曼编码

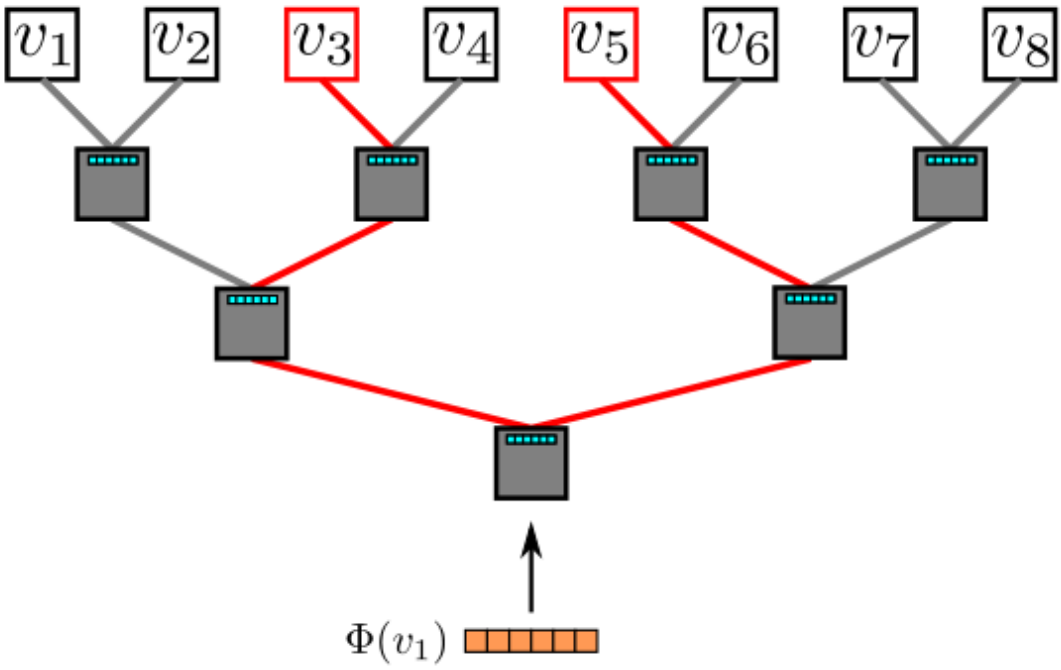
哈夫曼编码，左0右1，如图，D编码为0，B编码为10，C编码为110，A编码为111

Softmax其实就是多分类的Logistic Regression，相当于把很多个Logistic Regression组合在一起。Logistic Regression在这里的应用就是判断在哈夫曼树中走左子树还是右子树，其输出的值就是走某一条的概率。

应用到神经网络模型中，哈夫曼树的所有内部节点就类似神经网络隐藏层的神经元,其中，根节点的词向量对应我们的映射后的词向量，而所有叶子节点就类似于神经网络softmax输出层的神经元，叶子节点的个数就是词汇表的大小。在哈夫曼树中，隐藏层到输出层的softmax映射不是一下完成的，而是沿着霍夫曼树一步步完成的，因此这种softmax取名为"Hierarchical Softmax"。

在word2vec中，采用了二元逻辑回归的方法，即规定沿着左子树走，那么就是负类(霍夫曼树编码1)，沿着右子树走，那么就是正类(霍夫曼树编码0)。判别正类和负类的方法是使用sigmoid函数，使用霍夫曼树有两点好处，首先，由于是二叉树，之前计算量为V,现在变成了log2V。其次，由于使用霍夫曼树是高频的词靠近树根，这样高频词需要更少的时间会被找到，这符合贪心优化思想。

本文多层softmax



(c) Hierarchical Softmax.

图8 多层softmax

如图，树中的叶子节点均为图中的节点，对于每个内部节点，都利用一个二分类器来决定走哪个路径，如计算v3节点的概率，只需要经计算计算从根节点到v3节点上每一个路径的概率即可。由于二叉树的最长路径为O（log_2n），所以节点的计算时间为O（log_2n）。

本文Hierarchical Softmax概述部分参考

1.https://zhuanlan.zhihu.com/p/56139075

2.https://www.cnblogs.com/pinard/p/7243513.html

如果额头终将刻上皱纹，那么我们只能做到，不让皱纹刻在心上。（来源于《中国合伙人》）