DeepWalk原理与实践

DeepWalk是KDD 2014年的文章，论文题目是：Online learning of Social Representations。它属于Graph Embedding里比较经典算法。理清DeepWalk里的工作原理，可以更好的理解Graph Embedding其他文章。

# 简介

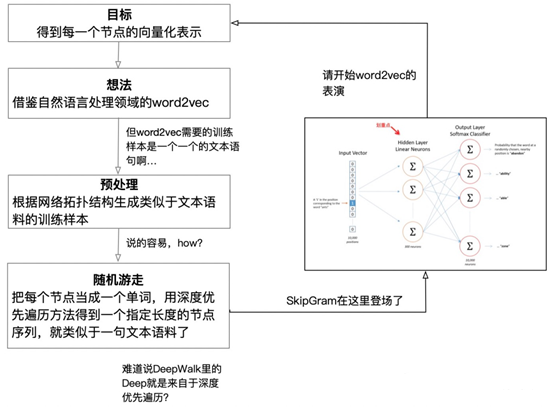
DeepWalk是一种学习图结构中节点的潜在表示的方法，这些潜在的表示法在连续向量空间中进行编码，编码到向量空间后节点的关系很容易被统计模型利用，两个点在图中共有的邻近点（或者高阶邻近点）越多，则对应的两个向量之间的距 离就越短。DeepWalk将随机游走得到的节点序列当做句子，从截断的随机游走序列中得到网络的局部信息，再通过局部信息来学习节点的潜在表示。DeepWalk是一种在线、可扩展、并行的学习方法，很适合应用到节点分类、异常检测等场景。

作者受自然语言处理领域word2vec的启发，思考在图结构中，是否可以根据图中节点与节点相邻关系，来学习每个节点的表示。论文中提出，在图结构中，以任意一个节点为起始节点，进行随机游走，当游走到最长的步数时，可以获取一串由节点构成的序列，这个序列就可以类比自然语言中的句子，节点类比句子中的词，然后利用 word2vec 的方法对每个节点进行表示学习。

|  |
| --- |
| Word2vec是一个非常基础和著名的词表示方法，利用句子中词与词之间的共现或相邻关系，对词进行表示学习，所学习到的词向量能准确的刻画词与词之间的实际意义关系。**想更深入理解word2vec的细节，可以参考：https://mp.weixin.qq.com/s/OhE5Za3FmnP9eEFq87brDg。** |

因为节点在是在图中随机游走的，所以DeepWalk是一种无监督的特征学习方法，具体来说，就是从可截断的随机游走中得到一串节点序列，利用 word2vec 方法学习每个节点的表示向量。我们认为这样学习到的表示向量可以捕捉到节点之间的邻近相似关系以及其所属社区（类别）的关系。

一言以蔽之，Graph Embedding是Word2Vec方法在图结构上的大胆尝试，将图上的结点转为低维向量表示。而DeepWalk是在物品组成的图(结点，邻接矩阵，权重矩阵等)中随机游走获得序列数据，对序列数据做word2vec获取embedding。



# 前置知识

## 图的定义

以社交网络中多分类任务为例：

G = (V, E)

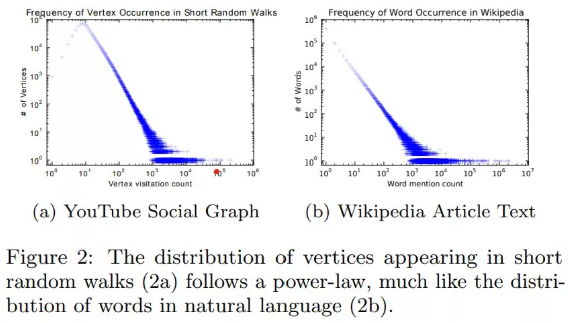
V表示图中的节点，节点的含义为图中的类别(member)。E表示图中的边，，表示一个带标签的图，其中， S表示节点的特征空间大小，，表示类别个数。

在实际应用中，是需要根据业务的具体逻辑去构建图。传统的机器学习方法是由X映射到Y，模型所需要学习的是如何准确的学习到这个映射函数。而本论文所提方法是独立于标签的分布信息，由图中的拓扑信息去学习节点的向量表示。是完全无监督学习方法。

该方法所学习到的表示向量可以使用在任何的分类算法中。

## 长尾分布

在论文中指的是power laws（幂律分布(度数大的节点比较少，度数小的节点比较多）），也称2-8定律，其实和长尾分布大同小异，在自然语言领域，我们发现大部分词的词频都很小，只有少数词的词频很高，符合长尾分布。而在YouTube Social Graph中，进行随机游走，发现节点出现频率的分布也是符合这种长尾分布（幂率分布）的，如下图所示：



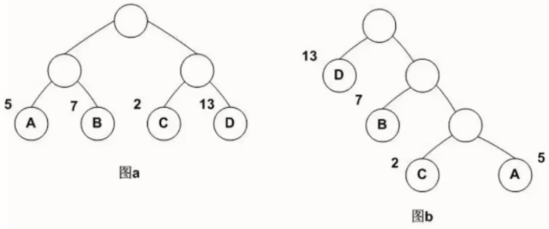
词频分布和节点频率分布一致，因此认为在自然语言处理领域内有效的word2vec方法可以复用在图结构中。

## Hierarchical Softmax

Hierarchical Softmax是word2vec中的一种采样算法。在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。

在DeepWalk生成的序列数据，产生节点Embedding的过程中也面临同样的问题，论文采用分层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



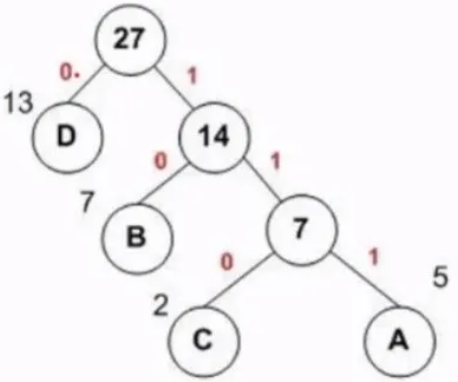
图中数字表示权重，表示节点重要程度。图a是常见的二叉树，图b就是图a转换过的最优二叉树。可以看出，D是最重要的，一个原则就是最重要的节点放在最前面，由此构造了图b的哈夫曼树。

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。



哈夫曼编码

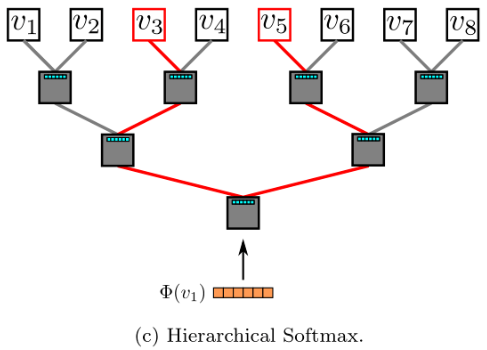
哈夫曼编码，左0右1，如上图，D编码为0，B编码为10，C编码为110，A编码为111。

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"。

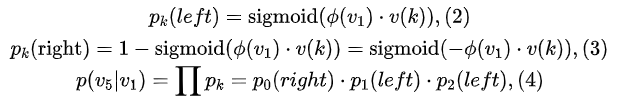
在word2vec中，采用了二元逻辑回归的方法，即规定沿着左子树走，那么就是负类(霍夫曼树编码1)，沿着右子树走，那么就是正类(霍夫曼树编码0)。判别正类和负类的方法是使用sigmoid函数，使用霍夫曼树有两点好处：首先，由于是二叉树，之前计算量为V，现在变成了log2V；其次，由于使用霍夫曼树是高频的词靠近树根，这样高频词需要更少的时间会被找到，这符合贪心优化思想。

### Deep Walk算法中的Hierarchical Softmax



如上图，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(log2n)，所以节点的计算时间为O(log2n)。

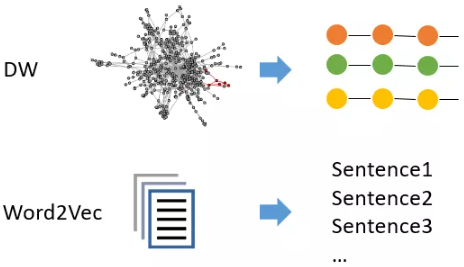
具体如何计算？我们以上图中用v1预测v5为例进行介绍。树的根部输入的是v1的向量，用表示。在二叉树的每一个节点上都存放一个向量，需要通过学习得到，最后的叶子节点上没有向量。显而易见，整棵树共有|V|个向量。规定在第k层的节点做分类时，节点左子树为正类别，节点右子树是负类别，该节点的向量用表示。那么正负类的分数如公式(2)(3)所示。在预测的时候，需要按照蓝色箭头的方向做分类，第0层分类结果是负类，第1层分类结果是正类，第2层分类结果是正类，最后到达叶子节点v5。最后把所有节点的分类分数累乘起来，作为v1预测v5的概率，如公式(4)所示，并通过反向传播进行优化。



顺便提一句，如果输出端是对所有词的softmax分类的话，那么在Skip-gram模型中，分别有输入和输出两个矩阵，一般是采用输出矩阵作为表示向量。但是如果采用Hierarchical Softmax分类的话，输出端就不存在输出矩阵了，就只能采用输入矩阵作为表示向量了。

## DeepWalk与Word2vec的关系

DeepWalk是采用随机游走的方式生成节点的序列，这些序列被看做一个个句子，序列中的节点对应句子中的单词，最终产生节点的Embedding还是使用word2vec得到的。



简单的说，DeepWalk = Random Walk + Sikp Gram。

# DeepWalk模型详解

## DeepWalk算法流程简述

* 获取用户行为序列
* 基于这些用户行为序列构建了物品相关图，图中的边是由用户行为产生的，比如为用户M先后购买了物品A和物品B，会产生了一条有向边由A指向B。其他的有向边也是同样的道理，如果有多个有A指向B的有向边，那么该条边的权重被加强。通过这样的方法将所有用户行为序列都转换成物品相关图中的边后，就得到全局的物品相关图。
* 采用随机游走的方式随机选择起始点，产生局部物品序列。
* 将这些物品序列当初句子进行word2vec建模，生成最终的物品Embedding向量。

## DeepWalk关键点详细剖析

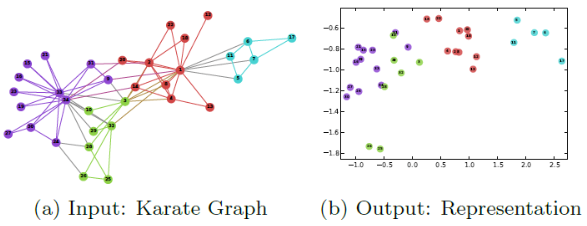
DeepWalk的关键在于如何利用网络生成语料库：给定一个流网络，从源节点出发，从其邻居中依概率选择下一步移动到的节点，概率以当前所在节点全部邻居权重归一化后来表示，概率越大到越容易被选择。重复这一选择过程，直至到达汇节点，则称完成一次随机游走。重复若干次这种随机游走就是我们需要的语料库。

对DeepWalk有了大概的认识之后，我们可以清晰的意识到，DeepWalk的关键技术点在于：

* 输入输出分别是什么？
* 如何获取物品组成的图？
* 怎么随机游走，获取序列？
* 怎么做embedding？

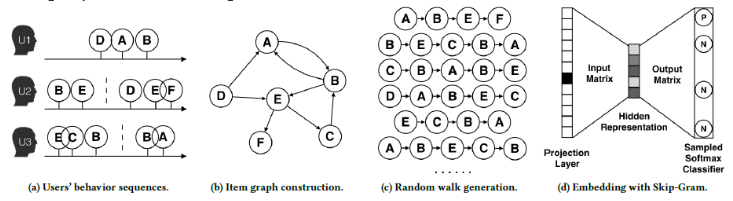
### 输入输出

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，论文的目的就是学习得到x的低维表示。从下图可以看到，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。



### 获取物品的图

物品之间的边连接是通过用户行为序列得出的，如下图所示。U2的序列被分成两个子序列，子序列内item的顺序就可以转为图中vertex的指向关系。总结全部用户的序列，item前后关系出现的次数也可以相对应地转为vertex的边权重。



Graph Embedding的一般流程

### 随机游走

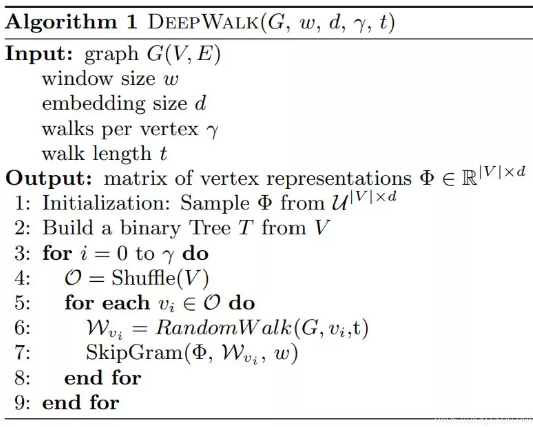
随机游走，顾名思义，就是从输入图中的任意一个结点vi开始，随机选取与其邻接的下一个结点，直至达到给定长度t，生成的序列。在论文中，对于每一个顶点vi，都会随机游走出条序列。

注：DeepWalk是无偏的，node2vec是有偏的。

采用随机游走有两个优点：

* 「利于并行化」：随机游走可以同时从不同的顶点开始采样，几个不同walker可以同时游走多条不同的线路，加快整个大图的处理速度；
* 「较强适应性」：从截断随机游走中获取信息，当图结构发生小的变化时，不需要重复重新的去学习，只需迭代地更新学习模型，就可以适应网络局部的变化，适合online learning。

DeepWalk论文中算法如下：



上图是算法的整个流程，3-9行为核心。

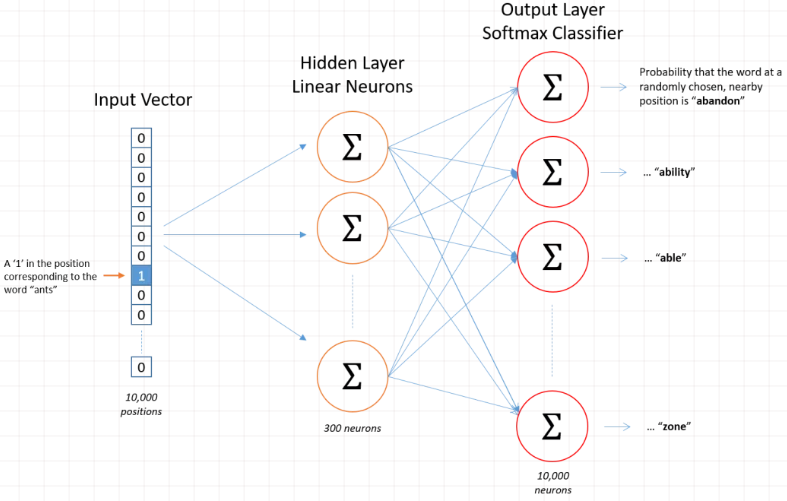
外循环规定了随机游走的次数，即将每次迭代都看作一次“传递”，并在每次传递过程中遍历一次节点，在开始遍历前，都会随机生成一个遍历顺序。最终对每个节点生成条节点序列。

在内循环中，遍历图上所有节点，生成一个节点序列，序列长度为t，通过skip-gram学习节点的嵌入。（特别注明，第4行的shuffle操作是方便后续的随机梯度下降。随机梯度下降的方法要求每次随机抽取一条样本来参与训练，而我们在遍历节点时，第一轮进行完，到第二轮时，如果还是以同样的节点顺序来遍历，那么样本的出现就具有了一定的规律，不够“随机”了，所以要对节点做一次随机排序，以加强样本顺序的随机性。）

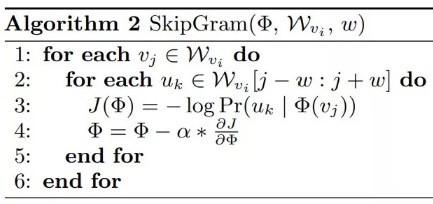
其中第2步构建二叉树的目的是为了方便后续Skip Gram模型的层次softmax算法。

### skip-gram作embedding

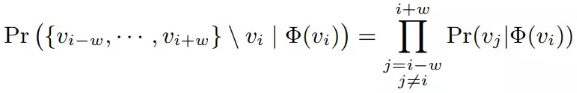
skip-gram是word2vec中的一种语言模型，定义一个滑动窗口，skip-gram会最大化单词同时出现在这个窗口的概率。



算法如下：



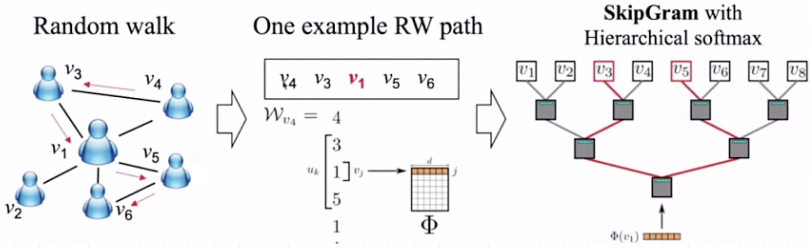
Skip-Gram算法表示由某个节点预测他周围的节点，是word2vec方法中的一种模型。



上述公式表示，由节点vi来预测与之相邻的周围节点vj，其中所得到的副产品这个 embedding矩阵就是我们要学习的目标。

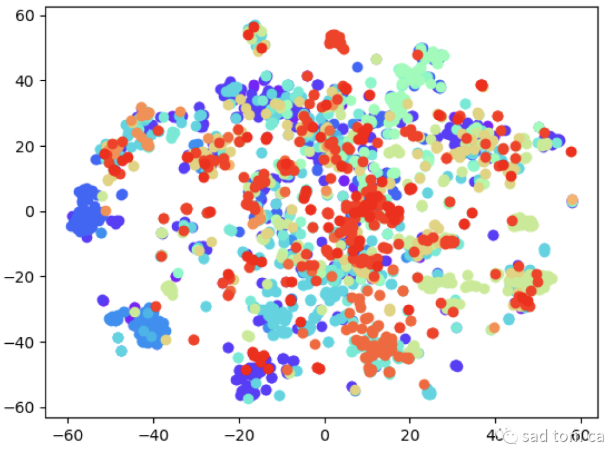
和自然语言处理相同，这里的skip-Gram同样面临计算复杂度很高的问题，如果不做任何处理，每次都需要对所有节点进行概率计算，时间复杂度过高，训练缓慢，由此引入了 Hierarchical Softmax。

Hierarchical Softmax是NLP中常用方法，其主要思想是以词频构建Huffman树，树的叶子节点为词表中的词，相应的高频词距离根结点更近。当需要计算生成某个词的概率时，不需要对所有词进行softmax计算，而是选择在Huffman树中从根结点到该词所在结点的路径进行计算，得到生成该词的概率，时间复杂度从O(N)降低到O(logN)（N个结点，则树的深度logN）。该方法同样适用在图结构的skip-Gram模型中。



Deep Walk算法的全过程

最后插一张图，是对DeepWalk的embedding结果做t-sne的结果图（颜色相同代表结点属于相同的类）。



|  |
| --- |
| 不管是在NLP中，还是在graph中，学习到的向量只是中间结果，用于作为下游任务的输入。例如在图中对节点做多标签分类任务时，第一步先通过DeepWalk进行无监督训练，得到所有节点的特征向量；第二步，通过一些分类器对节点进行分类。不同于传统的方法，DeepWalk将标签和表示空间分割开来，标签和表示向量相互独立，学习到的特征向量可以应用于各种不同的任务。而且试验证明，特征向量和最简单的分类算法相结合，比如逻辑回归，也能获得好的效果。 |

# DeepWalk的特点

## 适应性

在实际的社交网络中，由社交关系产生的图是不断发展变化的，当网络发生变化不能对整个网络重新进行计算，DeepWalk可以自适应的去学习，不必重复从头再学习。

## 社区意识

节点的潜在表示对应着维度空间中的距离，应该表示网络中对应的成员的相似度，以此保证网络的同质性。

## 低维度

当被标记的成员很少时，低维的模型一般表现的更好，并且收敛和推理速度更快。

## 连续性

需要通过图的潜在表示来对连续空间中的部分社区成员进行建模。除了提供对社区成员资格的细微视图之外，连续表示还可以使社区之间的决策界限平滑，鲁棒性更高。

# DeepWalk优缺点

优点：

* 支持数据稀疏场景，由于Skip Gram的特性，窗口内相邻的顶点会得到相似的一般表示，在网络标注顶点很少的情况也能得到比较好的效果
* 支持大规模场景（并行化）
* 可扩展，能够适应网络的变化

缺点：

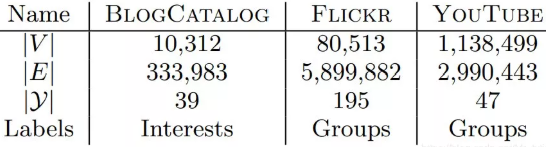
* 游走是完全随机的，但其实是不符合真实的社交网络的；
* 未考虑有向图、带权图

# 实验

## 数据集

论文中使用了三个数据集，数据集中包含用户，用户间的关系，以及兴趣标签或者用户组标签：

* BlogCatalog是博客作者的社交关系网络，标签代表作者提供的主题类别。
* Flickr是照片分享网站用户之间的联系网络，标签代表用户的兴趣组，如“黑白照片”。
* YouTube是流行的视频分享网站用户之间的社交网络，这里的标签代表喜欢不同类型视频（例如动漫和摔跤）的观众群体。



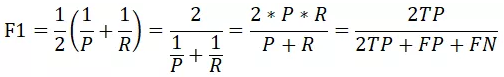
但是论文没有详细给出在三个数据集中，所构成的图结构中节点以及边的实际业务意义，例如在图中，每个数据集中节点具体指的是什么？以什么样的规则来定义边的？

## 评估指标

简单介绍下文中用于评估的几个指标（**要了解这两个评估指标的可以参考我之前总结的：https://mp.weixin.qq.com/s/K-2SCdW5POT7S0SCLrtBEw**）：

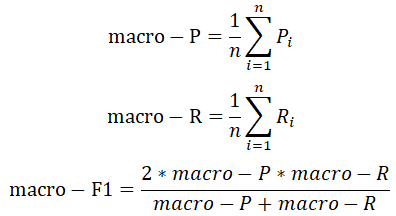
### F1

F1-score是精确率和召回率的调和平均值，其定义如下：



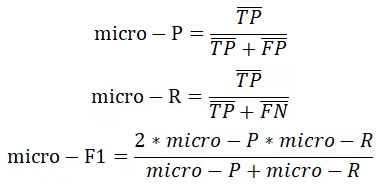
### macro-F1

先计算每个PR，取平均后，再计算F1。这样得到的就是宏查准率（macro-P）、宏查全率（macro-R），以及相应的宏 F1（macro-F1）



### Micro-F1

先计算混淆矩阵元素的平均，再计算 PR 和 F1。这样得到的就是微查准率（micro-P），微查全率（micro-R），微F1（micro-F1）



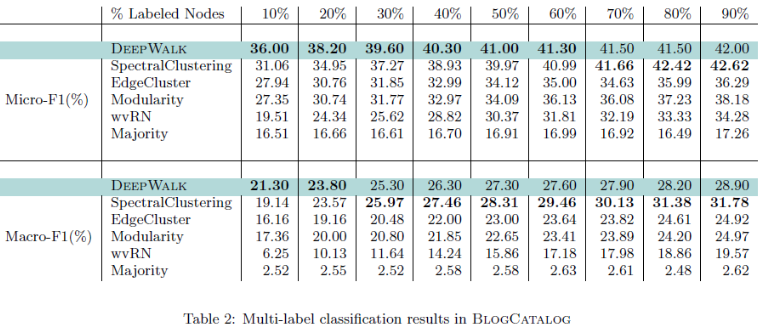
## 对比算法

* SpectralClustering
* Modularity
* EdgeCluster
* wvRN
* Majority

实验中通过多标签分类任务来评估算法的性能。从数据集中随机抽样标记节点的一部分(TR)，并将其用作训练数据。其余的节点被用作测试。重复这个过程10次，并报告Macro-F1和Micro-F1的平均性能。在训练集中，以上面所说的方式（DeepWalk + Skip-Gram + Hierarchical softmax）学习每个节点的表示向量，然后以学习到的表示向量作为特征，LR分类器进行训练，训练出一个分类模型；再在测试集上进行测试。将Deep Walk中的参数设置为：γ=80，w=10，d =128。在SpectralClustering，Modularity，EdgeCluster中，将维度设置为500。

## 实验结果

### BlogCatalog



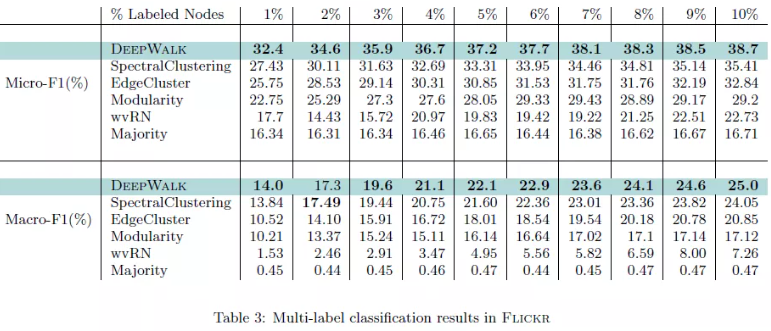
在实验中，将BlogCatalog网络上的训练比率(TR)从10％提高到90％，粗体数字表示每列中最高的性能。

结果分析：

* DeepWalk的性能始终优于EdgeCluster，Modularity和wvRN。DeepWalk在只有20％的节点被标记时的性能，比这些方法在90％的数据时被标记的情况下执行得更好。
* SpectralClustering的性能更具竞争力，但是当Macro-F1(TR≤20％)和Micro-F1(TR≤60％)上的标记数据稀疏时，DeepWalk仍然表现优异。

通过以上两点可以看出，算法的优势在于，只有小部分图表被标记时，具有强大的性能。

### Flickr

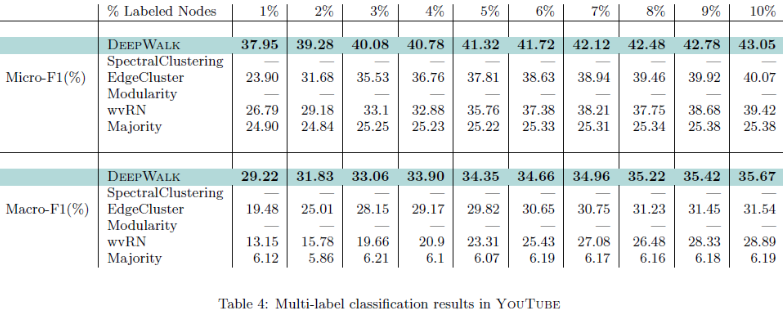


在实验中，将Flickr网络上的训练比率(TR)从1％变为10％。这相当于在整个网络中有大约800到8000个节点标记用于分类。上表给出了实验结果，粗体数字表示每列中最高的性能。

结果分析：

* 对于Micro-F1，DeepWalk的性能至少要比其他算法高出3％。DeepWalk可以比其他算法少60％的训练数据。
* 在Macro-F1中表现也相当不错，SpectralClustering最接近它。

### YouTube

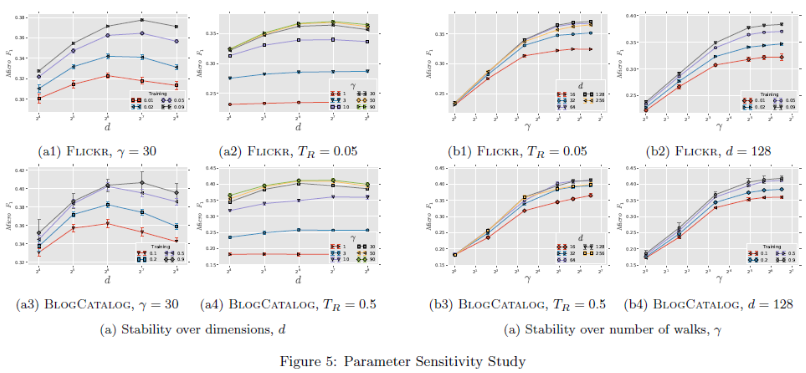


YouTube网络规模大，更接近现实世界网络。SpectralClustering和Modularity不能用于这种规模的网络。在实验中，训练比率(TR)从1％变化到10％，粗体数字表示每列中最高的性能。

结果分析：从实验中可以看出，大部分情况下DeepWalk都要优于传统算法。在达到相同的效果时，DeepWalk 相比传统算法所需要的有标签训练数据更少。在大规模网络上，有些传统算法已经跑不动了，DeepWalk也有不错的表现。并且在这样一个稀疏标记的环境中执行得非常好。

## 参数敏感度实验

为了评估DeepWalk的参数变化如何影响其在分类任务上的性能，我们对两个多标签分类任务(Flickr和BlogCatalog)进行了实验。实验中固定窗口大小和步长(w = 10，t = 40)，然后，改变嵌入维度(d)，每个游走长度(γ)，以及可用的训练数据量(TR)，以确定它们对网络分类性能的影响。



图(a)显示了维度变化的影响。图(a1)和(a3)分析了维度和训练数据比例对性能的影响。图(a2)和(a4)展示了维数和随机游走长度对性能的影响。得到如下结论：

* a1和a3显示模型的最佳维度取决于训练数据的数量。
* 通过a2和a4可以看出，在γ确定的情况下，不同维度下，算法的性能是相对稳定的。当γ>=30时，算法的性能比较好。两个图在γ的不同值之间的相对差异是一致的。

这些实验表明，算法可以生成各种大小的有用模型。模型的性能取决于随机游走的数量，而模型的最优维度取决于可用的训练样例。

图(b)显示了改变γ对性能的影响。结果对于不同的维度(图(b1)，图(b3))和不同训练数据量(图(b2)，图(b4))非常一致。增加γ对结果有很大的影响，但是当γ>10时，这种影响迅速减慢。这些结果表明，当随机游走的数量足够多时，我们才能够学习结点的有意义的潜在表示。

# 实践

参考：<https://github.com/jpegbert/NLP_Coding/tree/master/deep_walk1>

包含多种DeepWalk的实现方式

# 总结

本文所提方法不难，但是能借此应用到图结构中的确很有趣和有意义，使得word2vec这种简单有效的方法能应用的更加广泛。可以说DeepWalk给网络学习方向打开了一个新思路。

参考：

<https://mp.weixin.qq.com/s/CQaG7a1E9xGhUCygR9dEhA>

<https://mp.weixin.qq.com/s/ka9UL40szdecettjWCsAWg>

<https://mp.weixin.qq.com/s/w3zUlb4fdzF0jSRxdB9sUQ>

<https://mp.weixin.qq.com/s/gsxE1V_fNTDjIMhs3ZW72Q>

<https://mp.weixin.qq.com/s/ewTtyhkOISCN1JpMmvILGA>

<https://mp.weixin.qq.com/s/Td1IzeIKg48QQ0LoHJDA2g>

<https://mp.weixin.qq.com/s/SEXKZc7_YRyidq8CYxq4-w>

<https://mp.weixin.qq.com/s/6_ppfLzxExR35YqEw1nLHQ>

<https://mp.weixin.qq.com/s/VwKVFuKNBTpAsx50KQxELg>

<https://mp.weixin.qq.com/s/h1vDImYTLEheatZnScZwbg>

<https://mp.weixin.qq.com/s/m4J-2Ic5Gb476zQMaFBnkQ>

<https://mp.weixin.qq.com/s/QvPB3wNT3IeFuqLB8_nxsw>

<https://mp.weixin.qq.com/s/yR1Obbc5A_aWKfuGp1Vdmg>

<https://mp.weixin.qq.com/s/cz6_wu62GQonbP1AEHIrPg>

<https://mp.weixin.qq.com/s/fPlZyr2FobxEdekKexk72A>