Node2vec理论与实践

Node2vec是斯坦福大学的同学 2016 年发表于 ACM 的论文《node2vec: Scalable Feature Learning for Networks》。在这篇论文中作者提出了一个半监督学习算法—Node2Vec，采用了有偏的随机游走算法并结合Skip-gram算法学习Network Embedding，Node2Vec可以通过参数设置来控制搜索策略，从而有效的平衡了Embedding的同质性和结构有效性。

# 背景

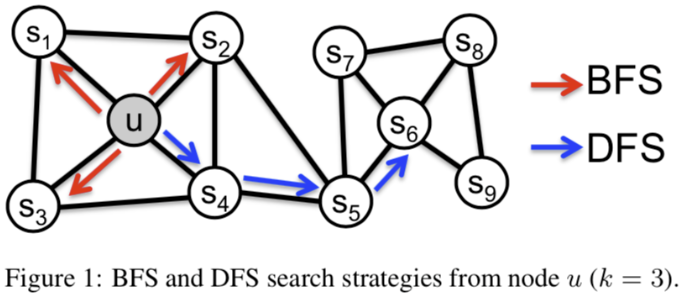
# 简介

Node2Vec是Stanford在2016年发表的论文。

Node2vec是Embedding技术在图中的应用，是DeepWalk的升级版（**要详细了解DeepWalk可以查阅：**）。Deepwalk可以认为是random walk + skip-gram，random walk本质上是一个dfs的过程，丢失了bfs的邻居结构信息。在论文中多次提到DeepWalk产生词序列的过程是僵硬、死板的，原因是DeepWalk的训练结果是依靠随机游走产生的序列的质量，没有解释节点间的关系。而node2vec可以简单理解为对deepwalk的随机游走过程进行优化，通过调节参数可以使用随机游走偏向bfs或dfs，从而使随机游走产生的词序列隐含节点之间的关系，主要是同义性（homophily）和同构性（structural equivalence）。与DeepWalk一样，Node2vec也是产生词序列，最终训练Embedding的过程仍然是skip-gram。

# 前置知识

论文中提出了两种度量节点相似性的方式，这里先探讨下，以更好的理解Node2vec的细节。



图中的结点大致可以分为两个community：一个是u,s1,s2,s3,s4组成的，以u为中心；另一个是由s5,s6,s7,s8,s9组成的，以s6为中心。

## 同义性

同义性，也称为同质性、内容相似，具有直接链接关系的两个节点，我们可以认为是内容相似的。例如上图中的s1和u。

同义性强调一个community里结点间的高度相互连接。例如图中的结点u和结点s1。通过广度优先搜索（BFS）获取的随机游走序列，可以突出结点间的同义性关系（简单理解，使劲转悠还是困在一个community里）。在经过skip-gram之后，同义性高的结点在特征空间里，距离要更近。

## 结构相似

同构性强调结点在网络结构中表现出网络拓扑结构组成上是类似的特点。例如图中的结点u和结点s6。通过深度优先搜索（DFS）获取的随机游走序列，可以突出结点间的同构性关系（简单理解，游走的方向更容易跨越community）。在经过skip-gram之后，同构性高的结点在特征空间里，距离也应该更近。

## DFS 和 BFS

DFS为上图中蓝色路径，可以理解为获取全局信息；BFS为上图中红色路径，可以理解为获取局部信息。

# Node2vec模型

首先引入用于学习节点Embedding的Skip-gram算法，并给出目标函数为：



其中，u为节点，为节点u通过采样策略S得到的邻居，f(u)是一个映射矩阵，相当于 Work2Vec中的输入向量。

为简便起见，我们给出两个假设：

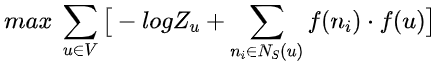
条件独立性假设：假设观察一个节点的领域与观察其他节点的领域相互独立，所以有：



特征空间对称型假设：在特征空间中，源节点与邻节点相互对称。所以有：



基于上面的假设，目标函数改为



其中，由于计算代价昂贵，所以我们用Negative Sampling进行优化。

## 搜索策略

由于网络是非线形的，所以我们需要一个策略来为Skip-gram提供一个线形的输入。一种常见的策略是通过游走的方式来对于给定源节点u的不同邻域进行采样，邻域不仅限与邻近的节点，而是与采样策略S有关。

在评价网络节点的相似性我们有同质性和结构等价性两个概念。

* 同质性是指同属于一个集群的两个节点更加相似，如下图的节点S1和节点u；
* 结构等价性是指两个具有相似结构的节点更加相似，如下图的节点S6和节点u。

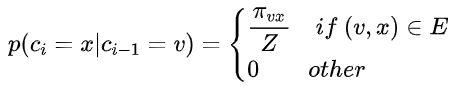
那么为了更能体现出网络的结构性，我们应该是用BFS采集还是DFS采集？这里可以简单思考一下。

答案是：BFS可以获得每个节点的邻居，强调的是局部微观视图，所以通过BFS采样的网络更能体现网络的局部结构，从而Embedding结果更能体现结构性；而DFS可以探索更大的网络结构，只有从更高的角度才能观察到更大的集群，所以其Embedding结果更能体现同质性。

这个答案是不是与我们的直觉有所相悖？

## 有偏的随机游走

先给出随机游走的公式：



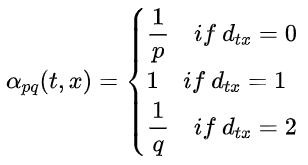
其中，ci表示第i次游走，表示节点x和节点v之间的未归一化之前的转移概率（即从节点v转移到节点x的概率），Z为归一化常数。

如果需要产生有偏的随机游走，一个比较简单的方法是令，但这样就没法适应不同网络结构，也不能引导我们的程序去探索不同类型的网络邻居。另外这里有偏的随机游走策略应该是统筹BFS和DFS的，以平衡同质性和结构等价性。

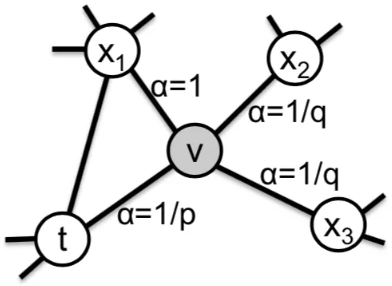
所以我们定义了一个非标准的转移概率 ：



其中：



其中，表示节点t和节点x之间的最短路径，取值为{0, 1, 2}。



结合下图来理解：

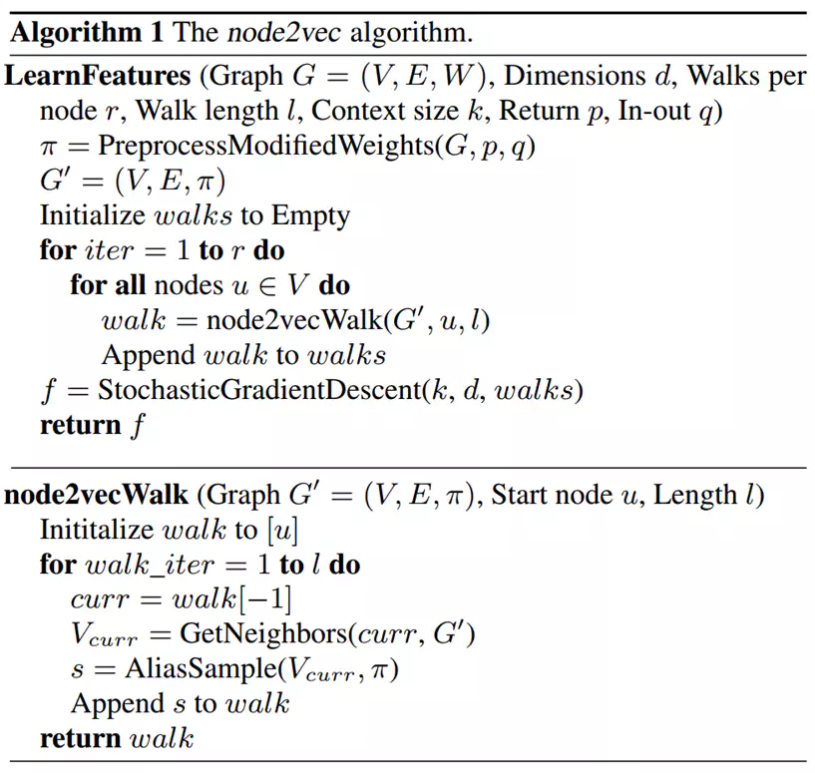
假设随机游走刚通过节点t来到节点v，现在考虑接下来的转移概率，节点x1与节点 t 的最短路径为1，所以，节点x2与节点t的最短路径为2，所以，同理节点x3。

直觉来看，参数p和q可以用来引导游走，近似地在BFS和DFS之间穿插，从而反映出不同节点在同质性和结构性上的亲和力。

* Return parameter p：参数p允许搜索程序重新访问遍历过的节点，其值越高，越不可能搜索已访问过的节点；
* In-out parameter q：参数q允许搜索程序区分“向内”和“向外”的节点。如果 q > 1 则，倾向于访问靠近节点t的节点，类似于BFS策略；反之，倾向于访问远离节点t的节点，类似于DFS策略。

最初的随机游走算法由于要存储所有的边，所以空间复杂度为O(|E|)，而有偏置的随机游走空间复杂度为：，其中a是网络节点的平均度数，其值通常非常小。

Node2Vec的随机游走方法兼容了DFS和BFS的优点，并且具有较低的时间复杂度和空间复杂度。下面是Node2Vec的伪代码：

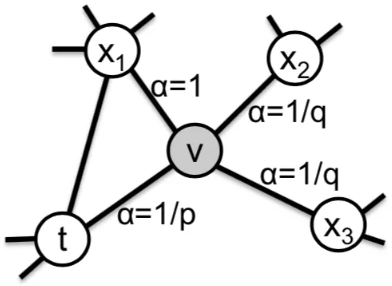


Node2Vec的算法共分为三个部分：预处理计算转移概率（PreprocessModifiedWeights，这个可以提前计算好），有偏置的随机游走（Node2VecWalk，加权采样使用的是Alias算法，其时间复杂度为O(1)）和异步随机梯度下降。每个阶段都可以并行化处理，这有助于加速 Node2Vec算法的训练。

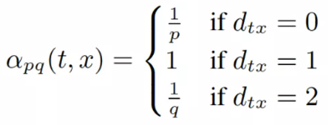
### Node2vec原理

Node2Vec通过对同义性和同构性的定义，区分出结点间的关系。那么在随机游走的过程中，样本侧重不同的结构特点，最终就能获得不同的embedding表达。所以接下来，要约定侧重的“力度”。

BFS和DFS产生的随机游走序列按照一定比例混合在一起，作为训练样本，确实可以影响模型的训练方向。但需要注意的是，在实际情况中，产生的序列并不是这么完美的，每个序列可能都会是BFS和DFS的混合结果。Node2Vec采用了名叫“搜索偏差”（Search bias）的量影响游走方向，记作。



上图是游走的序已经列从点t移动至点v，现在点面临的是四个选择，{t,x1,x2,x3}。接下来是分类讨论。



对上面公式的详细解释：

* 点t，即t和x相等。若从点v重新返回至点t，这是一种“重返”行为，由重返参数（Return Parameter）p控制，此时的搜索偏差；点v的前后两个点都是点t，它们的最远路径长为0。如果p越小，游走的序列里BFS的成分就越浓。
* 点x1，即t与x相连。若从点v移动至点x1，点x1与点t也有边连接，它们的最短路径长为1。此时的搜索偏差。
* 点x2和点x3，即t与x不相连。若从点v移动至点x2或者点x3，点x2、点x3与点t都没有边连接，它们的最短路径长为2。此时的搜索偏差。q被定义为出入参数(In-Out Parameter)。如果q越小，游走的序列里DFS的成分就越浓。

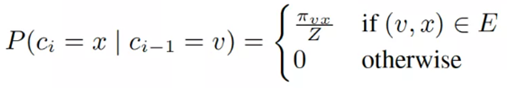
上述参数的意义解释如下：

* 参数：表示节点之间的最短路径，取值为0,1,2
* 参数p：返回参数，控制重新采样上一步已访问节点的概率。
* 当参数p>max(q,1)时，接下来采样的节点很大概率不是之前已访问节点，这一策略使得采样偏向dfs；
* 当参数p<max(q,1)时，接下来采样的节点很大概率是之前已访问节点，这一策略是的采样偏向bfs；
* 参数q：出入参数，控制采样的方向。
* 当参数q>1时，接下来采样的节点倾向于向t靠近，偏向于bfs；
* 当参数q<1时，接下来采样的节点倾向于向t远离，偏向于dfs；

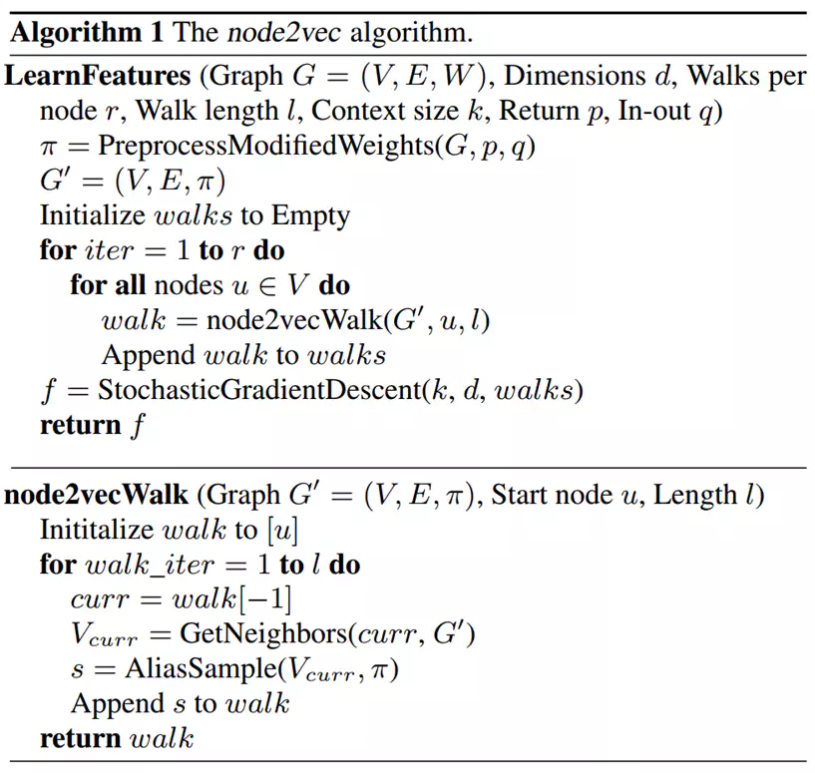
可以发现，当p=q=1时，node2vec就是DeepWalk。

上述过程在给定当前结点时，要根据前一个结点和后一个结点的最短距离给出搜索偏差；搜索偏差和边权重w相乘给出转移概率。

将转移概率归一化，得到最终的转移概率。



其中，表示节点x和节点v之间的未归一化概率（即从节点v转移到节点x的概率），Z为归一化常数。

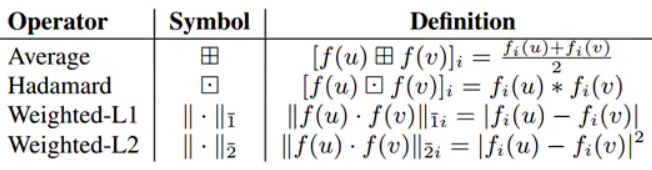


综上来看，这个转移过程是一个2阶马尔科夫过程。经过若干轮的循环遍历，Node2Vec就产生若干个序列，输入给skip-gram模型，得出结点的Embedding表示。

### Edge embedding

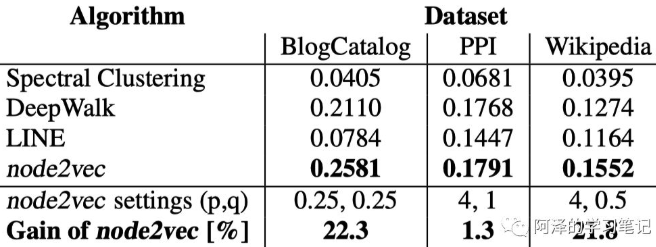
论文还有一个创新的地方在于：除了原本的 Embedding 任务还给出了边的预测的任务。

对于两个节点u和v，其Embedding向量表示是f(u)和f(v)，可以映射一个函数表示边的Embedding：g(u,v)，例如。其中，是映射后节点对(u,v)的Embedding维度。可选的操作有：

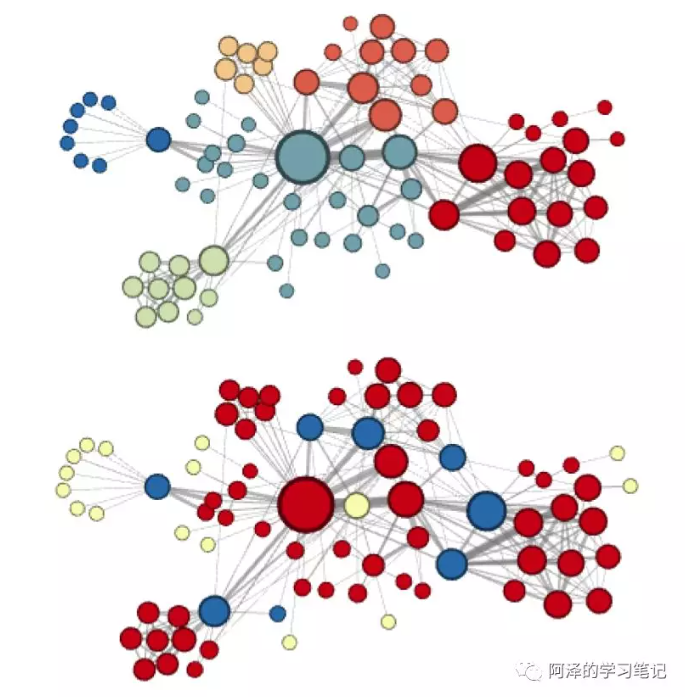


# 实验

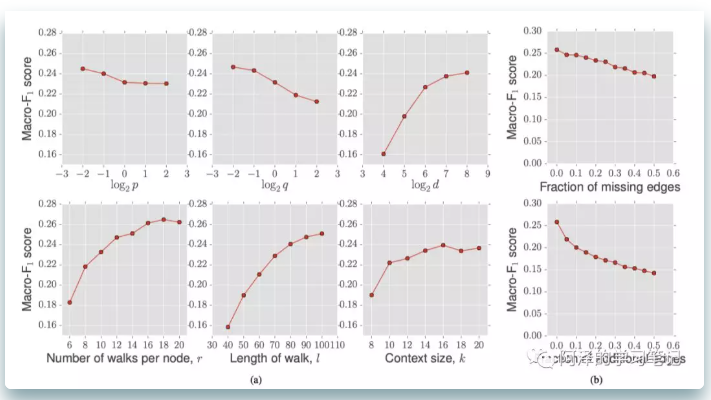
简单看一下实验，对比的实验除了DeepWalk、LINE外，还有谱聚类，实验目标是多分类，评价指标为Macro-F1和Micro-F1：



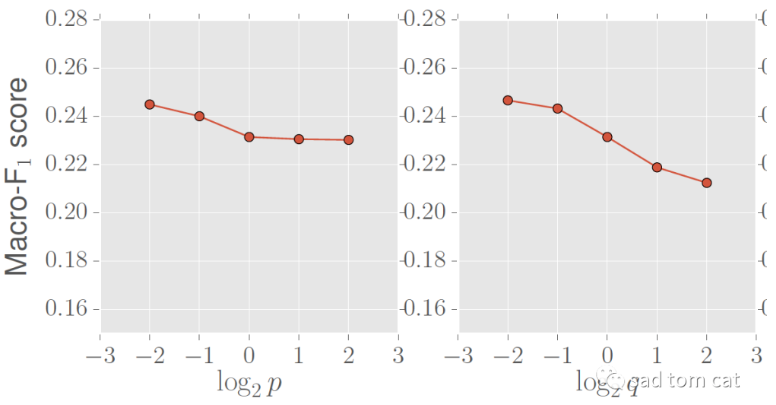
来看下Node2Vec关于同质性和结构等价性的效果，下图为Node2Vec生成《悲惨世界》共现网络的可视化图，标签颜色可以反映同质性（上）和结构等价性（下）。图片的上半部分参数值为 p=1, q=0.5，图片的下半部分参数值为 p=1, q=2。



## 参数敏感度



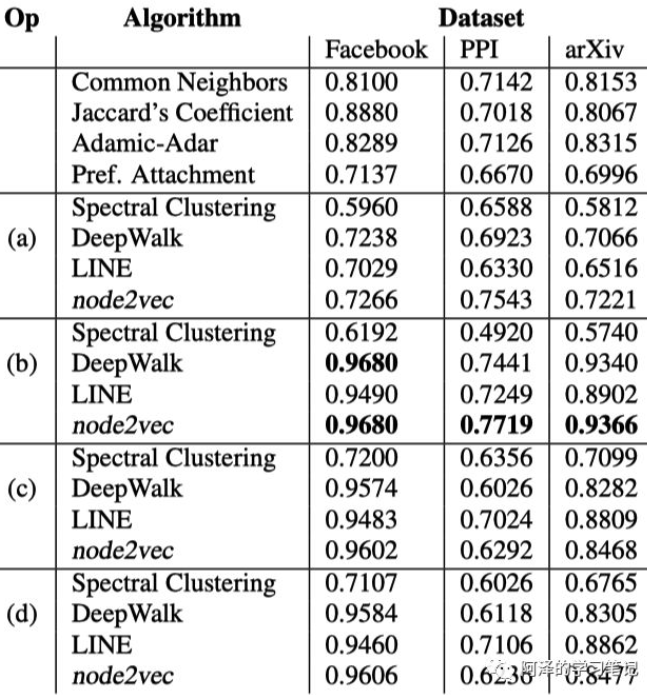
笔者只截取了p和q的参数敏感度比较，如下图所示：



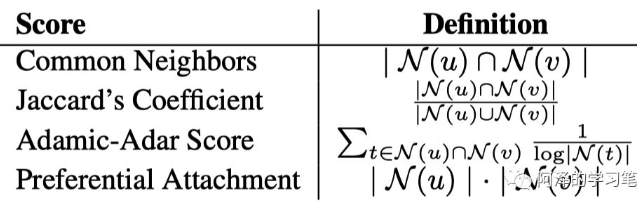
在公开数据集BlogCatalog上，以(多标签分类里有用到)为指标，使用不同的参数，讨论对模型效果的影响。显而易见，不管是增大p，还是增大q，模型的效果都变得更差了。“具体问题具体分析”——针对这个数据集，p小（高重返，偏向BFS）或者q小（高跳出，偏向DFS）对模型分类的结果更友好。

那为什么没有一个折中的参数组合呢？可惜的是原文也没有给出和组合的网格搜索结果。感兴趣的读者朋友可以试试。

## 边的预测

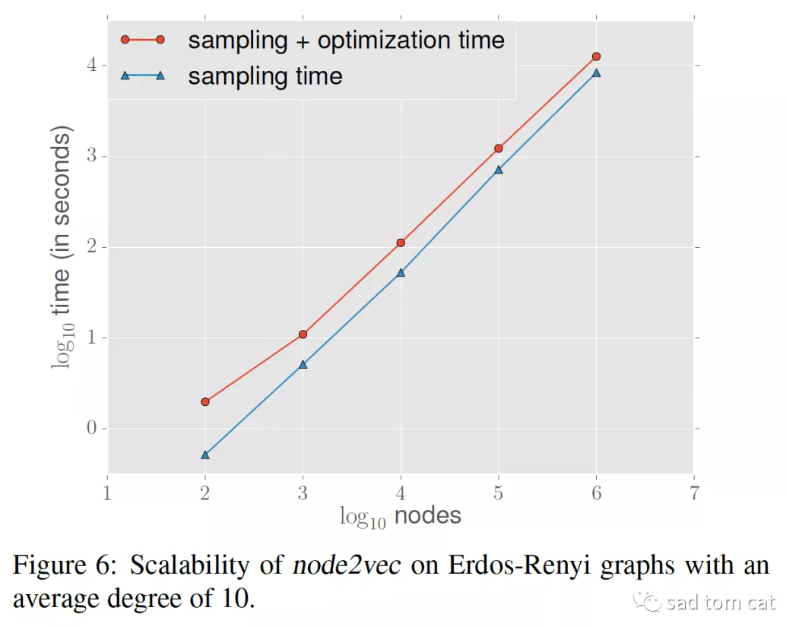


上图的第一行是一些指标，用于评测数据集，评测方法如下：



## 可扩展性

此处讨论的是结点个数对模型运行时间的影响。下图是在Erdos-Renyi图上依次采取100到1000000个结点（按10的倍数取），对模型运行时间的影响。



这里只分析红色的折线，在这个图上，算上采样和skip-gram训练模型的时间，时间对数和结点数对数基本上是线性的关系，给有意使用Node2Vec的人心里有个底。

# 实践

# 总结

Node2Vec是一个新的NetWork Embedding算法，其综合BFS和DFS优缺点，提出了有偏的随机游走算法，给随机游走的过程提供了一个可控的方向，并且为它的控制做出了合理的解释，最终的实验表明其具有良好的性能和可伸缩性。

从这篇文章中我们可以学到了：

* 一种新的Network Embedding算法——Node2Vec；
* 有偏的随机游走算法；
* BFS和DFS采样方法带来的结构性和同质性；
* 一种边预测的判定条件。

参考

<https://mp.weixin.qq.com/s/1P7VTCQAZdsDCpXQOUQYgg>

<https://mp.weixin.qq.com/s/u8WZqlsIcGCU5BqPH23S4w>

<https://mp.weixin.qq.com/s/mMOTsWiDfB40TG3HGW-z0g>