《Node2Vec：Scalable Feature Learning for Networks》

摘要：

1. **本文的背景：**自从 DeepWalk 论文提出了利用 Random Walk 来随机生成序列化数据供给 Skip-gram 模型做类似词嵌入的图节点嵌入训练后，图表示学习领域的研究不断取得重大进展。而随着实验的进一步深入，研究者们也发现了 DeepWalk 存在一定的局限性，为此本文的作者们在 DeepWalk 基础上又提出了一种更为有效的方法。
2. **本文的贡献：** 提出了一种名为 Node2Vec 的**有偏随机游走算法**来生成图网络中的序列化数据进行图嵌入。并且，首次将图嵌入方法应用到了**连接预测**任务之中。
3. **主要创新点：**通过使用有偏的二阶随机游走策略来产生序列化数据。更详细地说，通过设置超参数 p，q 用有弹性、有偏的随机游走策略有效平衡了来自 local（BFS）和 global（DFS）的节点网络结构信息。
4. **实验结果：**证明了 Node2Vec 在来自不同领域的几个真实世界网络中的**多标签分类**和**连接预测**任务上都明显优于现有的最先进的技术。

Node2Vec 的优化目标：

假设$f(u)$是将顶点u映射为 embedding 向量的映射函数，而对于图中的每个顶点$u \in V$，我们定义 $N_s(u) \in V$为通过采样策略S采样得到的顶点u的近邻顶点集合。因此，Node2Vec 优化的目标就是在给定每个顶点的条件下，令其近邻顶点出现的概率最大。即：

$$\max_f \sum_{u \in V} logPr(N_s(u)|f(u))$$

而为了方便上述公式易于处理，论文提出了两个标准假设：

（1） **条件独立性假设**——假设在给定源顶点的情况下，其近邻顶点出现的概率与近邻集合中其余的顶点无关。即：

$$Pr\big(N_s(u)\big|f(u)\big) = \prod_{n_i \in N_s(u)} Pr\big(n_i\big|f(u)\big)$$

（2） **特征空间对称性假设**——一个顶点在作为源顶点和作为近邻顶点时将共

享同一套 embedding 向量。即：

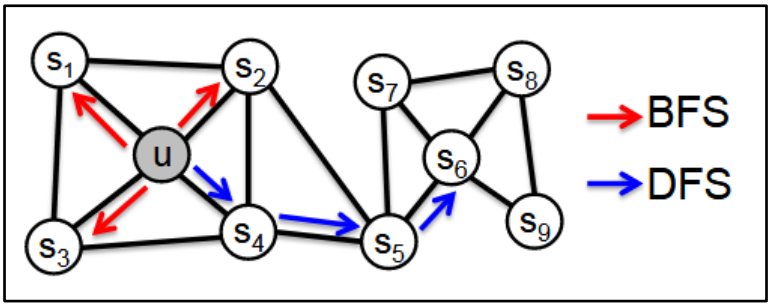
$$Pr(n_i|f(u)) = \frac{\exp\left(f(n_i)\cdot f(u)\right)}{\sum_{v\in V}\exp\left(f(v)\cdot f(u)\right)}$$

因此，根据以上的两个假设条件，最终的目标函数可以简化为：

$$\max_f \sum_{u\in V}\left[-logZ_u + \sum_{n_i\in N_s(u)} f(n_i)\cdot f(u)\right]$$

其中，配分函数（归一化分母）$Z_u = \sum_{v\in V}\exp\left(f(v)\cdot f(u)\right)$。而由于$Z_u$的计算代价过高，所以在实际训练中会采用负采样方法对其进行优化。

Node2Vec 的采样策略S：



如图所示，深度优先（BFS）和广度优先（DFS）策略代表着两种最极端的采样策略，其中 DFS 能体现网络节点的同质社群特征（homophily community），而 BFS 能体现网络节点的功能角色（结构）特征（structural equivalence）。但是，在真实世界中的网络往往是这两种特征的相互融合。

因此，Node2Vec 通过引入两个超参数 p 和 q 来控制随机游走策略，使其能够更好地应用到不同的网络图之中。

给定当前顶点v，访问下一个顶点x的概率为：

$$P(c_i = x|c_{i-1} = v) = \begin{cases}\frac{\pi_{vx}}{Z}, & if(v,x)\in E\\ 0, & otherwise\end{cases}$$

其中，π_{vx}是节点v和x之间的非规范化转移概率，Z是归一化常数。而在论文中，作者通过采用二阶随机游走对π_{vx}进行了定义，即：

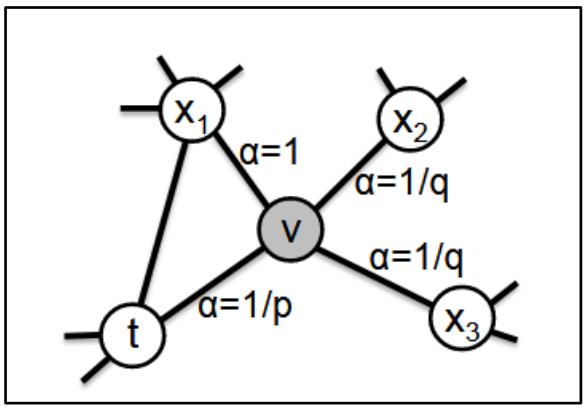
$$\pi_{vx} = \alpha_{pq}(t,x)\cdot w_{vx}$$

其中，w_{vx}代表连接边的连接权重（无权图则默认$w_{vx}=1$），$\alpha_{pq}(t,x)$代表平滑搜索策略，具体公式如下：

$$\alpha_{pq}(t,x)=\begin{cases}\frac{1}{p}, & if\ d_{tx}=0\\ 1, & if\ d_{tx}=1\\ \frac{1}{q}, & if\ d_{tx}=2\end{cases}$$

其中，t代表上一节点，x代表下一节点，p代表返回参数，q代表进出参数，d_{tx}代表顶点t和顶点x之间的最短路径距离。

超参数p和q对有偏游走策略的影响：



如上图所示，它描述的是当从上一节点t访问到当前节点v时，决定下一个访问节点x时每个顶点对应的α。而从前面公式中可知，α受超参数p和q的控制，具体而言：

• Return parameter，p

参数p控制着重复访问刚刚访问过的顶点t的概率。并且，注意到p仅作用于$d_{tx}=0$的情况，而$d_{tx}=0$代表着下一顶点x就是访问当前顶点v之前刚刚访问过的顶点t。因此，若p较高，则访问刚刚访问过的顶点t的概率会变低，反之则会变高。

• In-out parameter，q

参数q控制着游走是向外还是向内的，若$q>1$，则随机游走倾向于访问和t接近的顶点（更偏向于 BFS）。若$q<1$，则随机游走倾向于访问远离t的顶点（更偏向于 DFS）。

总之，通过调节p和q，我们可以平滑地调整采样策略，以探索图网络的不同特性。

Node2Vec 算法实现：

Algorithm 1 The *node2vec* algorithm.

LearnFeatures (Graph $G = (V, E, W)$, Dimensions d, Walks per node r, Walk length l, Context size k, Return p, In-out q)
  π = PreprocessModifiedWeights(G, p, q)
  $G' = (V, E, \pi)$
  Initialize $walks$ to Empty
  for $iter = 1$ **to** r **do**
    for all nodes $u \in V$ **do**
      $walk$ = node2vecWalk(G', u, l)
      Append $walk$ to $walks$
  f = StochasticGradientDescent(k, d, $walks$)
  return f

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node u, Length l)
  Inititalize $walk$ to $[u]$
  for $walk\_iter = 1$ **to** l **do**
    $curr = walk[-1]$
    V_{curr} = GetNeighbors($curr$, G')
    s = AliasSample(V_{curr}, π)
    Append s to $walk$
  return $walk$

Node2Vec 的算法伪代码如上图所示，其中u表示生成序列的起始节点，t代表上一节点，v代表当前节点，x代表下一节点，l代表随机游走序列的节点个数（长度），$N_S(t)$代表上一节点的邻居节点，k代表从$N_S(t)$中采样的节点个数，d代表嵌入向量的维度，p和q代表游走策略的超参数，r代表每个节点进行随机游走的次数。

整体的特征学习算法主要分为 3 步：

（1） 生成随机游走的采样策略；

（2） 对每个节点生成r个随机游走序列；

（3） 采用 Skip-Gram 训练得到节点的嵌入表示。

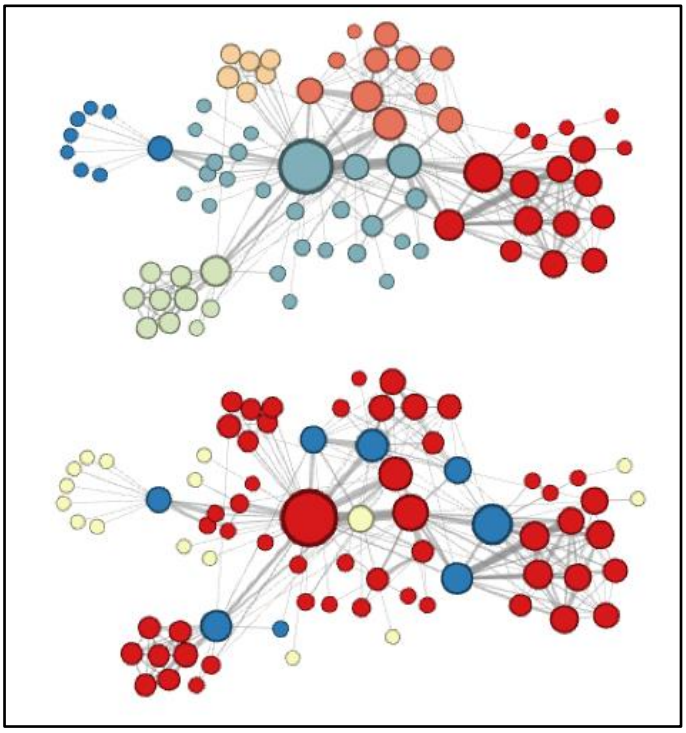
而在 Node2VecWalk 算法实现中，作者还用到了 **AliasSample 采样优化算法**，它的算法实现伪代码如下图所示。简单来说，Alias-Sample 通过利用内存空间（预处理，$O(n)$）来换取运行时间，使得采样的时间复杂度（效率）能够达到$O(1)$，大大节省了采样所需的时间。并且，它在大量反复抽样的情况下优势更为突出，能有效将离散分布的抽样转化为均匀分布的抽样。

Algorithm: Naive Alias Method

- **Initialization**:
  1. Multiply each probability p_i by n.
  2. Create arrays $Alias$ and $Prob$, each of size n.
  3. For $j = 1$ to $n - 1$:
     1. Find a probability p_l satisfying $p_l \leq 1$.
     2. Find a probability p_g (with $l \neq g$) satisfying $p_g \geq 1$
     3. Set $Prob[l] = p_l$.
     4. Set $Alias[l] = g$.
     5. Remove p_l from the list of initial probabilities.
     6. Set $p_g := p_g - (1 - p_l)$.
  4. Let i be the last probability remaining, which must have weight 1.
  5. Set $Prob[i] = 1$.
- **Generation**:
  1. Generate a fair die roll from an n-sided die; call the side i.
  2. Flip a biased coin that comes up heads with probability $Prob[i]$.
  3. If the coin comes up "heads," return i.
  4. Otherwise, return $Alias[i]$.

实验部分

1. 案例研究：《悲惨世界》网络

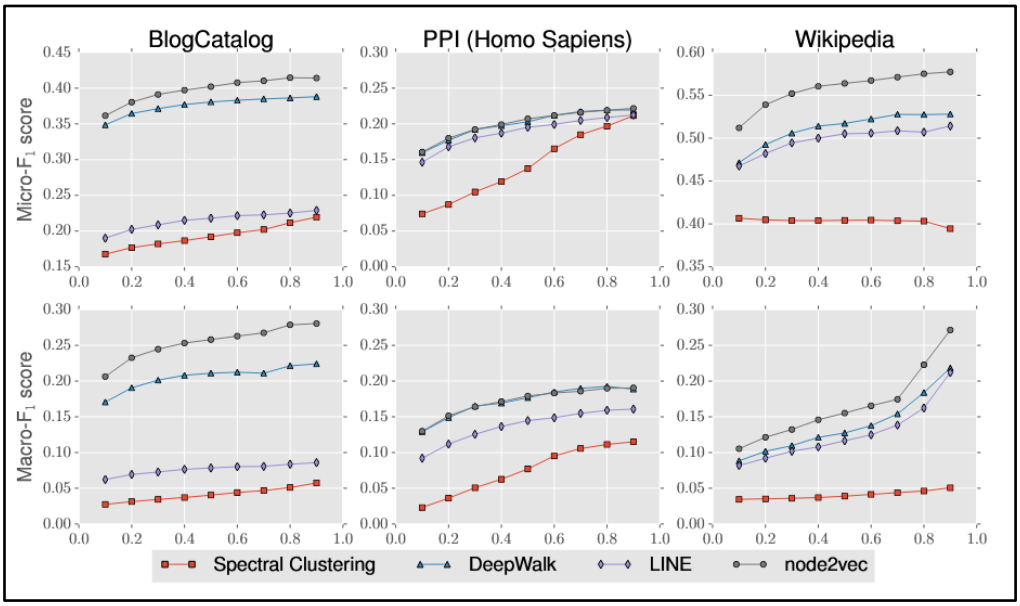


如图所示，这是雨果的《悲惨世界》的人物社交关系图。其中，顶部网络的节点颜色展示了节点之间的**同质社群特征**（对应超参数 p 大 q 小）；底部网络的节点颜色代表了节点之间的**功能角色特征**（对应超参数 p 小 q 大）。可见，不同的超参数 p 和 q 能发掘网络中不同的特征信息。

2. 多标签分类任务

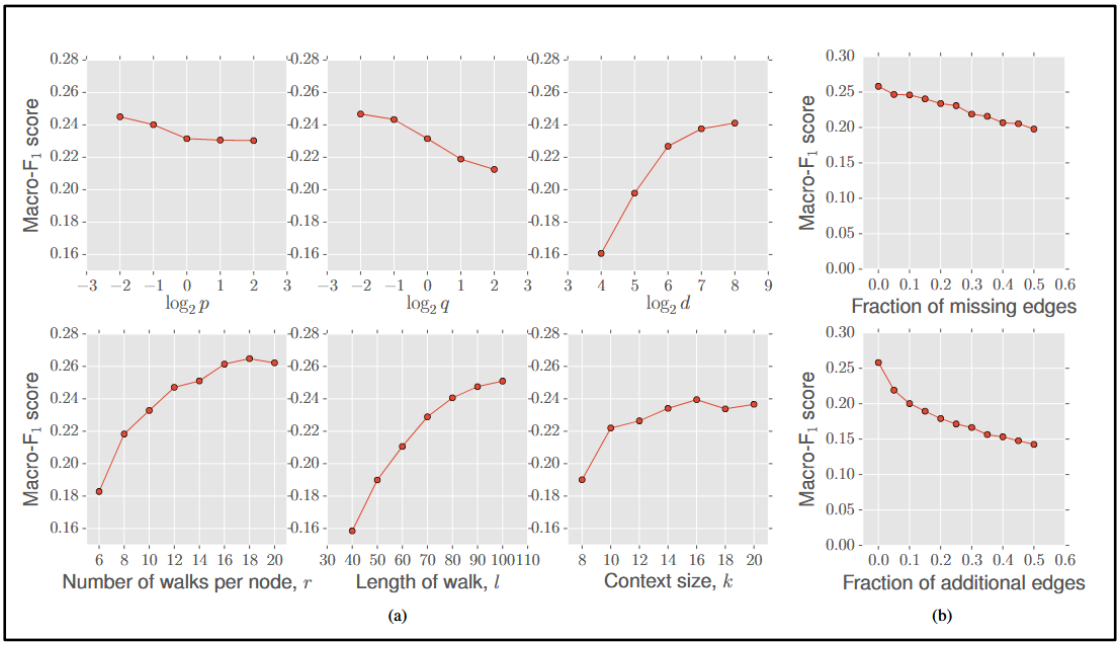
Algorithm	Dataset		
	BlogCatalog	PPI	Wikipedia
Spectral Clustering	0.0405	0.0681	0.0395
DeepWalk	0.2110	0.1768	0.1274
LINE	0.0784	0.1447	0.1164
node2vec	**0.2581**	**0.1791**	**0.1552**
node2vec settings (p,q)	0.25, 0.25	4, 1	4, 0.5
Gain of *node2vec* [%]	**22.3**	**1.3**	**21.8**

如图所示，在 Macro-F1 得分上，**Node2Vec 在不同的数据集上都取得了相比其他方法的最好成绩**。并且，在部分数据集上，Node2Vec 甚至能达到 20%的提升。



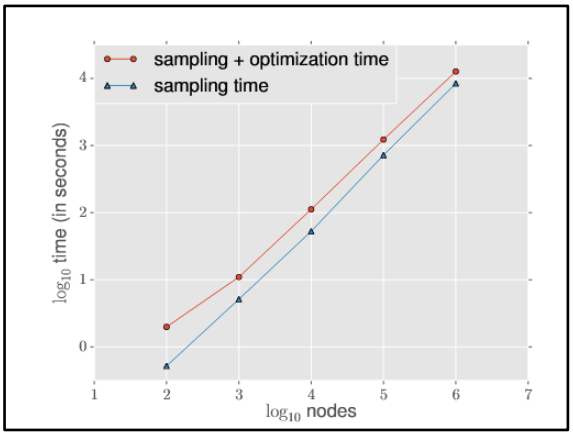
如图所示，不同训练的标记数据量对各算法的性能评估。其中，X轴表示标记数据的比例，而顶部和底部中的Y轴分别表示 Micro-F1 和 Macro-F1 分数。而从中我们可以发现，**DeepWalk 和 Node2Vec 在 PPI 上提供了相当的性能，但在所有其他网络中，Node2Vec 表现最佳**。此外，标记数据越多，算法效果越好。

3. 调参对比实验和扰动分析



如图所示，不同参数的大小都会影响最终的 F1 得分，而通过随机剔除连接和随机增加连接的实验（如图(b)所示），我们也可以发现 Node2Vec 算法的**鲁棒性较好**，能有效对抗扰动（缺失和噪声）。

4. 可扩展性



如图所示，随着网络中节点个数的增长，算法的运行时间是和节点数呈线性相关的。因此，Node2Vec 的算法**可扩展性较好**。

5. 连接预测

说明：在本篇论文中，作者通过利用定义的二元运算符来学习节点之间的边

特征。并且，还将这些方法与一些常见的启发式方法进行了比较。具体的二元运算符如下表所示：

Operator	Symbol	Definition
Average	$\boxplus$	$[f(u) \boxplus f(v)]_i = \frac{f_i(u)+f_i(v)}{2}$
Hadamard	$\boxdot$	$[f(u) \boxdot f(v)]_i = f_i(u) * f_i(v)$
Weighted-L1	$\|\cdot\|_{\bar{1}}$	$\|f(u) \cdot f(v)\|_{\bar{1}i} = \|f_i(u) - f_i(v)\|$
Weighted-L2	$\|\cdot\|_{\bar{2}}$	$\|f(u) \cdot f(v)\|_{\bar{2}i} = \|f_i(u) - f_i(v)\|^2$

Table 1: Choice of binary operators $\circ$ for learning edge features. The definitions correspond to the ith component of $g(u, v)$.

而常见的启发式方法如下表所示：

Score	Definition
Common Neighbors	$\mid \mathcal{N}(u) \cap \mathcal{N}(v) \mid$
Jaccard's Coefficient	$\frac{\mid \mathcal{N}(u) \cap \mathcal{N}(v) \mid}{\mid \mathcal{N}(u) \cup \mathcal{N}(v) \mid}$
Adamic-Adar Score	$\sum_{t \in \mathcal{N}(u) \cap \mathcal{N}(v)} \frac{1}{\log \mid \mathcal{N}(t) \mid}$
Preferential Attachment	$\mid \mathcal{N}(u) \mid \cdot \mid \mathcal{N}(v) \mid$

Table 3: Link prediction heuristic scores for node pair (u, v) with immediate neighbor sets $\mathcal{N}(u)$ and $\mathcal{N}(v)$ respectively.

而实验的最终结果如下：

Op	Algorithm	Dataset		
		Facebook	PPI	arXiv
	Common Neighbors	0.8100	0.7142	0.8153
	Jaccard's Coefficient	0.8880	0.7018	0.8067
	Adamic-Adar	0.8289	0.7126	0.8315
	Pref. Attachment	0.7137	0.6670	0.6996
(a)	Spectral Clustering	0.5960	0.6588	0.5812
	DeepWalk	0.7238	0.6923	0.7066
	LINE	0.7029	0.6330	0.6516
	node2vec	0.7266	0.7543	0.7221
(b)	Spectral Clustering	0.6192	0.4920	0.5740
	DeepWalk	**0.9680**	0.7441	0.9340
	LINE	0.9490	0.7249	0.8902
	node2vec	**0.9680**	**0.7719**	**0.9366**
(c)	Spectral Clustering	0.7200	0.6356	0.7099
	DeepWalk	0.9574	0.6026	0.8282
	LINE	0.9483	0.7024	0.8809
	node2vec	0.9602	0.6292	0.8468
(d)	Spectral Clustering	0.7107	0.6026	0.6765
	DeepWalk	0.9584	0.6118	0.8305
	LINE	0.9460	0.7106	0.8862
	node2vec	0.9606	0.6236	0.8477

Table 4: Area Under Curve (AUC) scores for link prediction. Comparison with popular baselines and embedding based methods bootstapped using binary operators: (a) Average, (b) Hadamard, (c) Weighted-L1, and (d) Weighted-L2 (See Table 1 for definitions).

可见，采用 Hadamard 方式的 Node2Vec 和 DeepWalk 算法均取得了最好的结果。

总结：

DeepWalk 存在的缺点：

（1） 用完全随机游走来训练节点嵌入向量（刚性策略效果并不好）。

（2） 仅能反映**相邻节点**的**社群**相似信息。

（3） 无法反映节点的**功能角色**相似信息。

Node2Vec 图节点嵌入算法：

- Node2Vec 解决了**图嵌入**问题，将图中的每个节点映射为一个向量 (嵌入)。
- 向量 (嵌入) 包含了节点的语义信息 (相邻社群和功能角色)。
- 语义相似的节点，向量 (嵌入) 的距离也近。
- 向量 (嵌入) 可用于后续的分类、聚类、Link Prediction、推荐等任务。
- 在 DeepWalk 完全随机游走的基础上，Node2Vec 增加了 p、q 参数，实现了有偏随机游走。而不同的 p、q 组合，对应了不同的探索范围和节点语义。
- DFS 深度优先搜索，相邻的节点，向量 (嵌入) 距离相近。
- BFS 广度优先搜索，相同功能角色的节点，向量 (嵌入) 距离相近。
- 而 DeepWalk 可以看成是 Node2Vec 在$p = 1, q = 1$的特例。

总之，相比其他算法而言（例如 DeepWalk 和 LINE），Node2Vec 中的搜索策略既灵活又可控，我们可以通过调节参数 p 和 q 来探索网络邻域，并且这些搜索参数具有更直观的解释性。此外，从实际的角度来看，Node2Vec 还是可扩展的，且对扰动也具有良好的鲁棒性。

对于本文的感悟：

总的来说，Node2Vec 是在 DeepWalk 的基础上发展出来的，并且成功地解决了 DeepWalk 所存在的一些缺陷，而这些有效的改进离不开作者们敏锐的洞察力和深入浅出的思考。从整体来看，Node2Vec 仅仅是对 DeepWalk 做出了一点优化，但优化的效果却好的出人意料。或许论文的创新点就是这样，不断地在前人的基础上发掘问题，解决问题，乃至更近一步、产生质变。