Node2Vec 论文+代码笔记

深度传送门 2020-06-03

以下文章来源于AINLP ， 作者太子長琴

AINLP

一个有趣有AI的自然语言处理社区：关注AI、NLP、机器学习、推荐系统、计算广告等...

作者：太子長琴（NLP算法工程师）

Paper：node2vec: Scalable Feature Learning for Networks

Code：aditya-grover/node2vec

核心思想：通过给网络节点的邻居定义一个灵活的概念，并设计了一个能够有效探索邻居多样性的有偏随机游走程序，来学习网络的节点表征。

What

动机问题

许多任务涉及图节点和边的分析。

- 任何有监督算法都需要包含信息、具有区分度的独立特征，一般的方法是利用专家知识在领域内做特征工程。
- 另一种方法是作为优化问题学习特征表征，但因为参数爆炸，训练时间长、复杂度高。
- 还有就是纯粹的无监督算法，但是当前的技术无法令人满意地定义和优化网络中的合理目标。经典方法基于线性和非线性维度缩减技术，它们通过转换一个网络数据表征矩阵，让数据表征的方差最大化来优化目标。所以，这些方法总是涉及到矩阵特征分解，这对于现实中的大型网络来说比较昂贵，而且，在各种图预测任务中表现不好。
- 或许还可以设计一个能够保留节点邻域的目标，用 SGD 优化，但是这依赖网络领域的严格概念，导致对网络特有的连接模式不敏感。具体来说，网络上的节点可以通过它们所属的社区（同形的）进行组织，其他情况下可以基于节点的结构角色组织。所以算法应该能够做到：
  - 来自同一网络社区的节点 embedding 足够接近
  - 担当相似角色的节点具有相似的 embedding

模型算法

$$\max_f \sum_{u \in V} \log \Pr(N_S(u)|f(u))$$

- G = (V, E)
- f → R, size |V | × d 表示节点特征表征向量
- Ns(u) ⊂ V 表示采样策略 S 下节点 u 的邻居

相关假设：

- 条件独立：

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

- 对称性：源节点和邻居节点在特征空间中彼此对称。

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

最后的损失函数：

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

其中，

$$Z_u = \sum_{v \in V} \exp(f(u) \cdot f(v))$$

BFS 和 DFS

网络中节点上的预测任务经常在两种相似性之间交织：同构和结构对等。

- 根据同构假设，高度互连并属于相似网络集群或社区的节点应紧密地嵌入在一起。
- 根据结构对等假设，在网络中具有相似结构角色的节点应紧密嵌入在一起。

通过 BFS 采样的邻域会导致与结构对等紧密对应的嵌入。在 DFS 中，采样的节点更准确地反映了邻域的宏观视图，这对于基于同构性推断社区至关重要。

> 这里稍微解释下，根据 BFS 采样，意味着只要节点邻域类似，节点就类似，这个距离可能很远，其结果就是结构对等的相似；根据 DFS 采样，意味着节点可能比较接近，属于同一社区，其结果就是同构相似。

node2vec

node2vec 通过一种灵活的偏向随机游走程序来抽样，可以同时以 BFS 和 DFS 方式探索邻域。

给定起始节点 u，ci 是随机游走的第 i 个节点，所以 c0 = u：

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

π_vx 是 v 和 x 之间未归一化的转移概率，Z 是归一化常数。假设刚从 t 节点转移到 v 节点：

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

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

d 表示节点间的最短距离，必须是 0 1 或 2。直觉上看，p 和 q 控制程序多快探索或离开 u 的邻居节点，具体的，允许程序近似地在 BFS 和 DFS之间插值。

- p 控制立刻回访一个节点的概率。
- q 允许搜索区分 “向内” 和 “向外” 节点。q > 1 倾向于在 t 附近游走（类似 BFS 行为），q < 1 倾向于远离 t（类似 DFS 行为）。

Algorithm

```
# 对所有节点执行 num_walks 轮随机游走
def simulate_walks(num_walks, walk_length):
    walks = []
    nodes = list(G.nodes())
    for walk_iter in range(num_walks):
        random.shuffle(nodes)
        for node in nodes:
            walks.append(node2vec_walk(walk_length=walk_length, start_node=node))
    return walks
# 节点随机游走（本文核心代码）
def node2vec_walk(walk_length, start_node):
    walk = [start_node]
    while len(walk) < walk_length:
        cur = walk[-1]
        cur_nbrs = sorted(G.neighbors(cur))
        if len(cur_nbrs) > 0:
            if len(walk) == 1:
                nxt = cur_nbrs[alias_draw(alias_nodes[cur][0],
                                          alias_nodes[cur][1])]
```

```
            else:
                prev = walk[-2]
                nxt = cur_nbrs[alias_draw(alias_edges[(prev, cur)][0],
                                          alias_edges[(prev, cur)][1])]
            walk.append(nxt)
        else:
            break
    return walk
```

随机游走得到的是节点序列，每个序列的长度为 `walk_length` ，共有 `num_walks * num_nodes` 个序列。所有的序列丢入 Word2Vec 模型训练即可得到 node 向量。

`alias_draw` 的实现比较简单：

```
def alias_draw(J, q):
    K = len(J)
    kk = int(np.floor(np.random.rand()*K))
    if np.random.rand() < q[kk]:
        return kk
    else:
        return J[kk]
```

可以看出其主要作用就是随机从 J 中取出一个元素。

重点是 `alias_nodes` 和 `alias_edges` ，二者分别遍历所有节点和边，为每个节点和边生成 alias。

```
def get_alias_node(node):
    unnormalized_probs = [G[node][nbr]['weight'] for nbr in sorted(G.neighbors(r
    norm_const = sum(unnormalized_probs)
    normalized_probs =  [float(u_prob)/norm_const for u_prob in unnormalized_prc
    return alias_setup(normalized_probs)
def get_alias_edge(src, dst, p, q):
    """
    src: t
    dst: v
    dst_nbr: x
    """
    unnormalized_probs = []
```

```
    for dst_nbr in sorted(G.neighbors(dst)):
        w_vx = G[dst][dst_nbr]['weight']
        if dst_nbr == src:
            unnormalized_probs.append(w_vx/p)
        elif G.has_edge(dst_nbr, src):
            unnormalized_probs.append(w_vx)
        else:
            unnormalized_probs.append(w_vx/q)
    norm_const = sum(unnormalized_probs)
    normalized_probs =  [float(u_prob)/norm_const for u_prob in unnormalized_prc
    return alias_setup(normalized_probs)
```

节点直接使用了权重（毋庸置疑）作为转移概率，边使用了上面的公式计算转移概率。关于 `alias_setup` 放在了最后面的 “附录” 部分，可以移步查看。

Edge Features

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 $g(u, v)$.

就是根据不同的操作方式定义了几种 “边” 的定义，label 就是边是否存在。

特点创新

主要贡献在于定义了节点网络邻居的一个灵活概念。通过有偏差的可以有效探索给定节点不同邻域的随机游走族来实现。

- 提出了 node2vec，一种网络特征学习的高效可伸缩算法，它使用 SGD 有效地优化了一种新颖的网络感知的邻域保留目标。
- 展示了 node2vec 如何符合网络科学中的既定原则，为发现符合不同等价关系的表示形式提供了灵活性。

- 将基于邻域保留目标的 node2vec 和其他特征学习方法从节点扩展到成对的基于边缘的预测任务的节点对。

- 根据经验评估 node2vec 的多标签分类和对几个实际数据集的 link 预测。

How

构造输入

首先需要通过节点和关系来构造图，输入很简单：n 行，每行分别是起始节点、终止节点、权重，输入数据默认是有向的。

```
G = nx.read_edgelist('data.edgelist', nodetype=int,
                     data=(('weight', float),), create_using=nx.DiGraph()) #
有向图
G = G.to_undirected()
```

这样就完成了图的构建。

开始训练

输入构造完后首先需要预处理，然后随机游走，最后利用 Word2Vec 进行训练。

- 预处理：这里主要是计算每个节点、边的归一化概率。节点的比较简单，直接计算的是邻居节点的归一化权重。边稍微复杂一些，这里也是论文的创新部分。具体做法如下：
  - 给定起始节点 t 和目标节点 v，v 的邻居节点为 x，权重 w_vx。
  - 如果 x 和 t 相同，π_vx = w_vx / p；
  - 如果 x 和 t 之间有边，π_vx = w_vx；
  - 其他情况：π_vx = w_vx / q。
  - 计算归一化转移概率。
- 随机游走：
  - 给定起始节点，不断 walk 选择下一个节点，直到节点序列长度达到设定的长度。
  - 对所有节点执行上步操作（即设置每一个节点为初始节点进行游走）。
  - 执行 n 轮游走。

- 训练：
  - 随机游走得到序列矩阵： `(num_walks * num_nodes) × walk_length` 。
  - 喂入 Word2Vec 模型进行训练。
  - 得到 Node 向量矩阵： `num_nodes × dimensions` 。

使用结果

训练完后得到的其实就是一个 Word2Vec 的模型，使用方法并无差别。

数据实验

数据处理和模型都不复杂，看看实验参数情况。

Multi-label classification

这里主要和三种方法做了对比：

- 光谱聚类：一种矩阵分解方法，将图 G 的标准化拉普拉斯矩阵的顶部 d 个特征向量用作节点的特征向量表示。
- DeepWalk：其实就是 Node2Vec 的特殊情况（当 p=1，q=1 时）。
- LINE：两阶段学习特征，在第一阶段，它通过 BFS 式的模拟在节点的临近节点上学习 d / 2 维；在第二阶段，它通过严格从源节点开始以 2 跳距离采样节点来学习下一个 d / 2 维。

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**

Table 2: Macro-F_1 scores for multilabel classification on BlogCatalog, PPI (Homo sapiens) and Wikipedia word cooccurrence networks with 50% of the nodes labeled for training.

这个 Gain of node2vec 意思是采用 pq 设置带来的增益。

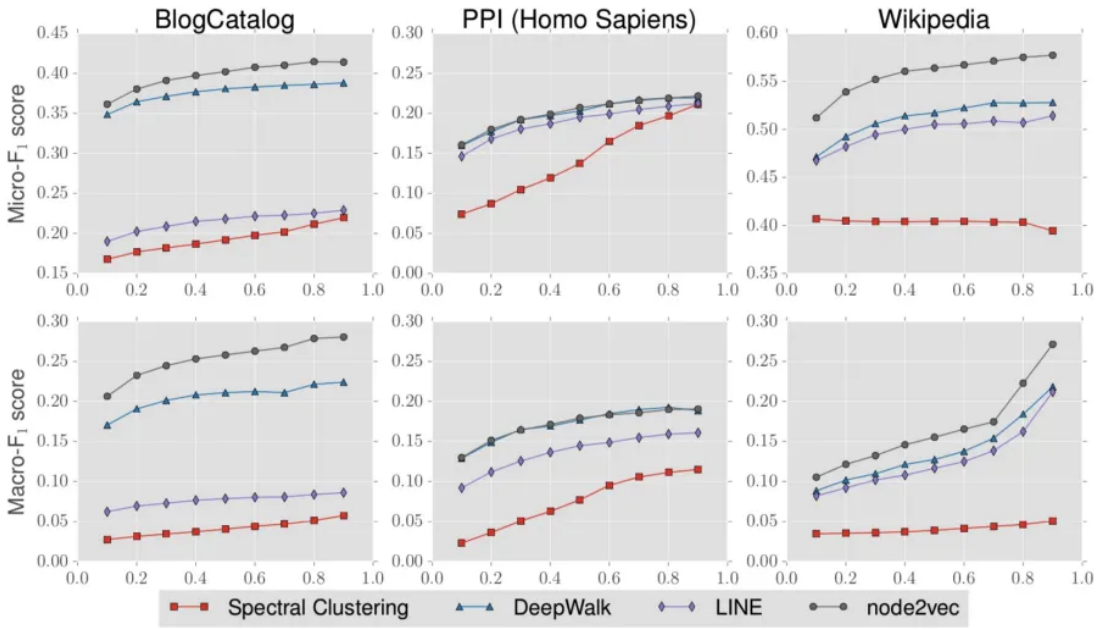

Figure 4: Performance evaluation of different benchmarks on varying the amount of labeled data used for training. The x axis denotes the fraction of labeled data, whereas the y axis in the top and bottom rows denote the Micro-F_1 and Macro-F_1 scores respectively. DeepWalk and *node2vec* give comparable performance on PPI. In all other networks, across all fractions of labeled data *node2vec* performs best.

这里的 x 轴表示训练-测试数据比例。

参数敏感性

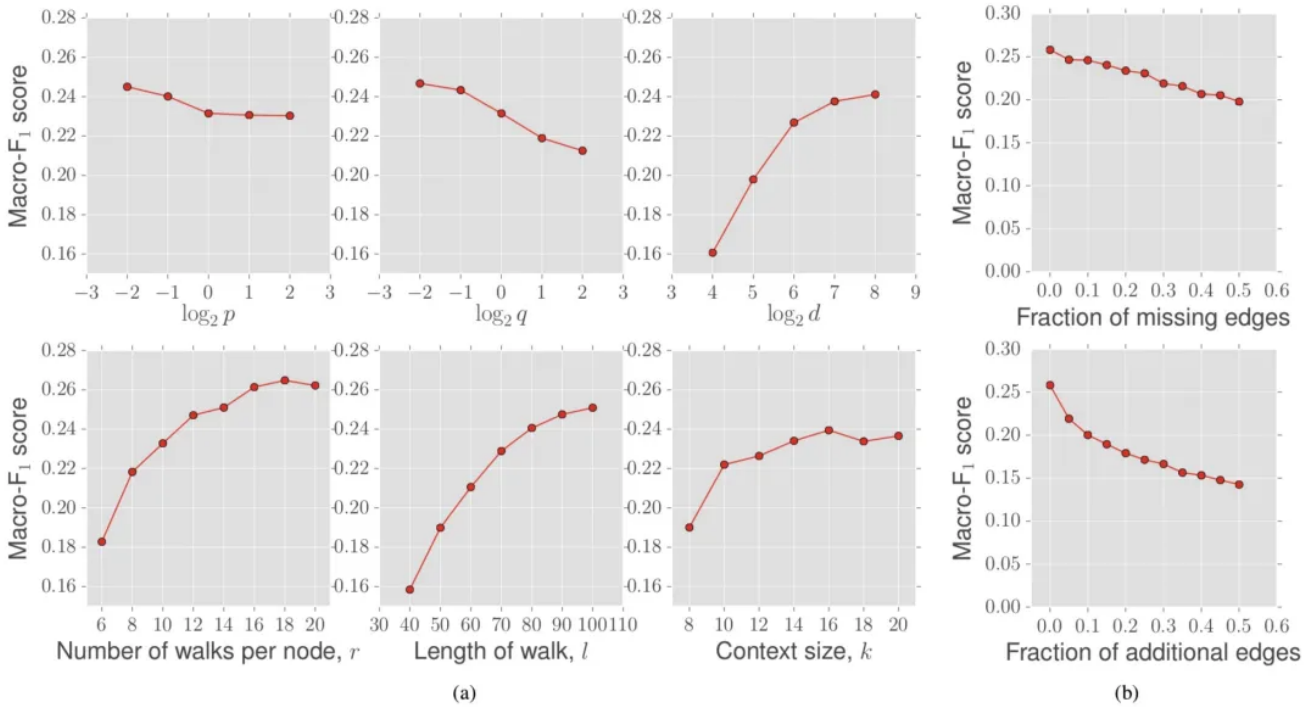

Figure 5: (a). Parameter sensitivity (b). Perturbation analysis for multilabel classification on the BlogCatalog network.

低 q 倾向于向外探索，低 p 倾向于在节点附近。d 包括 r, l, k。k 就是训练词向量时的窗口大小。最后一列的两幅图分别表示缺失边和随机边。

Link prediction

随机移除 50% 的边作为正例，负例随机抽样无边的节点对。

评分方法：

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.

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).

Discussion

相关工作

- 常规的特征工程基于特征提取技术（代表是基于网络属性的人工种子特征）
  - B. Gallagher and T. Eliassi-Rad. Leveraging label-independent features for classification in sparsely labeled networks: An empirical study. In Lecture Notes in Computer Science:

Advances in Social Network Mining and Analysis. Springer, 2009.

- K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos. It's who you know: graph mining using recursive structural features. In KDD, 2011.

- 无监督特征学习方法通常图的各种矩阵表示，特别是拉普拉斯算子和邻接矩阵。可以看做利用降维技术。然而这类方法在计算和统计上有性能问题（矩阵特征分解），另外这类方法优化的目标对网络中的不同模式不鲁棒，而且对网络和预测任务之间的关系做了假设。这样的假设在许多情况下是合理的，但不能有效地推广到各种网络。

  - M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In NIPS, 2001.
  - S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, 290(5500):2323–2326, 2000.
  - J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. Science, 290(5500):2319–2323, 2000.
  - S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin. Graph embedding and extensions: a general framework for dimensionality reduction. IEEE TPAMI, 29(1):40–51, 2007.

- NLP 领域的 Skip-gram Model 使用 SGD 和 Negative-Sampling，通过优化邻域保留似然目标来学习单词的连续特征表示。

  - T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In ICLR, 2013.
  - T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In NIPS, 2013.

- 受 Skip-gram 启发，最近的研究通过将网络表示为 "文档" 建立了网络的类比。可以从基础网络中采样节点序列，然后将网络转变为按顺序排列的节点序列。不同的采样策略对应不同的特征表征，但没有明确的获胜采样策略可在所有网络和所有预测任务中使用。这是先前工作的主要缺点，无法为网络中的节点采样提供任何灵活性。node2vec 通过设计一个不依赖于特定采样策略的灵活目标克服了这一限制，并提供了参数来调整探索的搜索空间。

  - B. Perozzi, R. Al-Rfou, and S. Skiena. DeepWalk: Online learning of social representations. In KDD, 2014.
  - J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. LINE: Large-scale Information Network Embedding. In WWW, 2015.

- 基于图的有监督深度学习框架使用多层非线性变换直接将下游预测任务的损失函数最小化。

  - K. Li, J. Gao, S. Guo, N. Du, X. Li, and A. Zhang. LRBM: A restricted boltzmann machine based approach for representation learning on linked data. In ICDM, 2014.
  - X. Li, N. Du, H. Li, K. Li, J. Gao, and A. Zhang. A deep learning approach to link prediction in dynamic networks. In ICDM, 2014.

- Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. In ICLR, 2016.
- F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu. Learning deep representations for graph clustering. In AAAI, 2014.
- S. Zhai and Z. Zhang. Dropout training of matrix factorization and autoencoder for link prediction in sparse graphs. In SDM, 2015.

特殊情况

- 当网络没有边时，全部为孤立点，无法学习表征。
- 当网络任何两个节点都相互连接时，为连通图，可以为每个节点表征。
- p 和 q 为 1 时，退化为 DeepWalk。
- p 很小时，游走局限在节点附近；p 很大时相反。
- q 很小时，倾向于向远处探索；q 很大时相反。

打开脑洞

开始胡思乱搞模式……

我：搞了半天突然发现本文的创新点概括成一句话就是对边的权重做了些微调整。怎么感觉如此 Fuck 简单呢？

大神：那你还想咋滴？创新可不就是那么一点点的修补吗？！要不你也来创新一个？

我：噢……（内心 OS：WTF，浪费了我一整天时间！）

仔细想想，感觉无论什么样的 node embedding，都是需要先获得一个序列，而很多算法也都在如何获得这个序列上下功夫。就直观感受来看，随机游走在图上显然是比较合适的方式，这其实也是一种随机采样。

词向量也是类似的方式，其实文字不也可以看作是语言的一种采样么？我们看到的每句话、每段文本，都可以看做是图中的一条路径，也可以看做一个采样序列。这样的话，我们看到的一篇文章其实就是一幅图，我们能否用这种图结构来表征文本？这样的图能否表现出某些自然倾向的特性？

Appendix

`alias_setup` 是一个用于从离散分布中进行非均匀采样的工具，参考自这里，不过我已经打不开了。它的输入是一个归一化的概率分布，具体实现如下：

```
def alias_setup(probs):
    K = len(probs)
    q = np.zeros(K)
    J = np.zeros(K, dtype=np.int)

    smaller = []
    larger = []
    for kk, prob in enumerate(probs):
        q[kk] = K*prob
        if q[kk] < 1.0:
            smaller.append(kk)
        else:
            larger.append(kk)

    while len(smaller) > 0 and len(larger) > 0:
        small = smaller.pop()
        large = larger.pop()

        J[small] = large
        q[large] = q[large] + q[small] - 1.0
        if q[large] < 1.0:
            smaller.append(large)
        else:
            larger.append(large)
    return J, q
```

和 `alias_draw` 联系起来就是根据概率分布取样：

```
res = []
probs = [0.1, 0.2, 0.7]
J, q = alias_setup(probs)
for i in range(100):
    choose = alias_draw(J, q)
    res.append(choose)
collections.Counter(res)
# 结果类似这样：Counter({2: 72, 1: 18, 0: 10})，大致接近 1: 2: 7
```

对了，random 的 choices 函数， `numpy` 的 random choice 函数均可以实现同样的功能：

```
collections.Counter(random.choices([0, 1, 2], weights=[0.1, 0.2, 0.7], k=100))
# Counter({2: 65, 1: 22, 0: 13})
collections.Counter(np.random.choice([0, 1, 2], size=100, replace=True, p=
[0.1, 0.2, 0.7]))
# Counter({2: 68, 1: 23, 0: 9})
```

所以，其实本文的实现可以简化，直接把每个节点和边的 `normalized_probs` 存下来就好了，随机游走代码可以改为这样：

```
choose = np.random.choice(list(range(len(cur_nbrs))),
                          size=1, replace=True, p=curr_normalized_probs)
nxt = cur_nbrs[choose[0]]
```

另外，随机游走时，起始节点我觉得可以不用处理，直接换成起始边就好了。也就是这样：

```
for walk_iter in range(num_walks):
    random.shuffle(edges)
    for edge in edges:
        walks.append(node2vec_walk(walk_length=walk_length, start_edge=edge))
```