DeepWalk论文笔记及应用

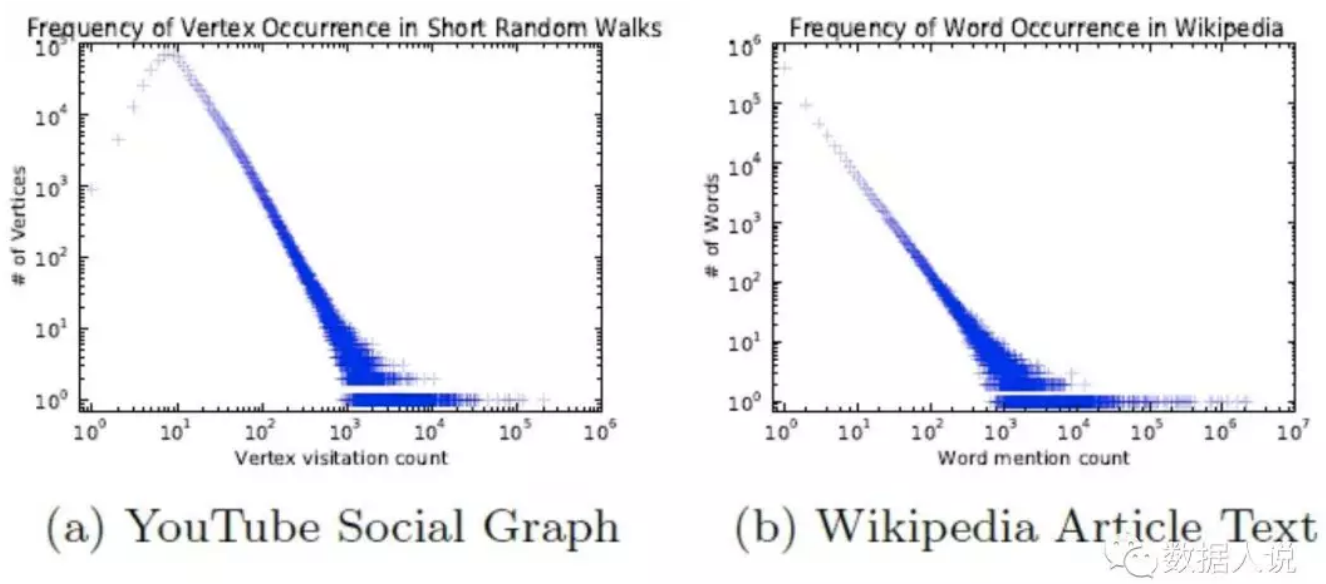
原创 Devin Jiang 数据人说 2020-02-01

背景

DeepWalk模型算是网络表示学习（Network Embedding）的开山之作，其思想来源于之前介绍的Word2Vec。本文从理论和应用方面做了一些复盘。

1. 原理介绍

论文提出了一种网络嵌入的方法叫DeepWalk，思想来源于Word2Vec，作者首先介绍了自然语言中的词频满足统计学中的幂律分布（Power Laws），而在网络中对每个顶点进行指定深度的随机游走得到的顶点序列也同样满足幂律分布，如下图所示。因此可以相应地将NLP中的word2vec运用在图中的顶点表示上，故而有了DeepWalk算法。



power-law

DeepWalk输入是一张图或者网络，输出为网络中顶点的向量表示。DeepWalk通过截断随机游走(truncated random walk)学习出一个网络的社会表示(social representation)，由于SkipGram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。并且该方法还具有可扩展的优点，能够适应网络的变化。

DeepWalk 的算法描述，DeepWalk包括两个部分，一个是随机游走生成器，一个是参数更新程序

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 $\Phi \in \mathbb{R}^{|V| \times d}$

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

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

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

其中第2步是构建Hierarchical Softmax，第3步对每个节点做γ次随机游走，第4步打乱网络中的节点，第5步以每个节点为根节点生成长度为t的随机游走，第7步根据生成的随机游走使用skip-gram模型（可见word2vec模型）利用梯度的方法对参数进行更新。

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。

2. 原论文复现

对理论大致了解后， 可以直接运行下面[1]中的代码，按照下面的执行命令就可以复现论文的结果，需要耐心等待下，下图是我之前做的笔记

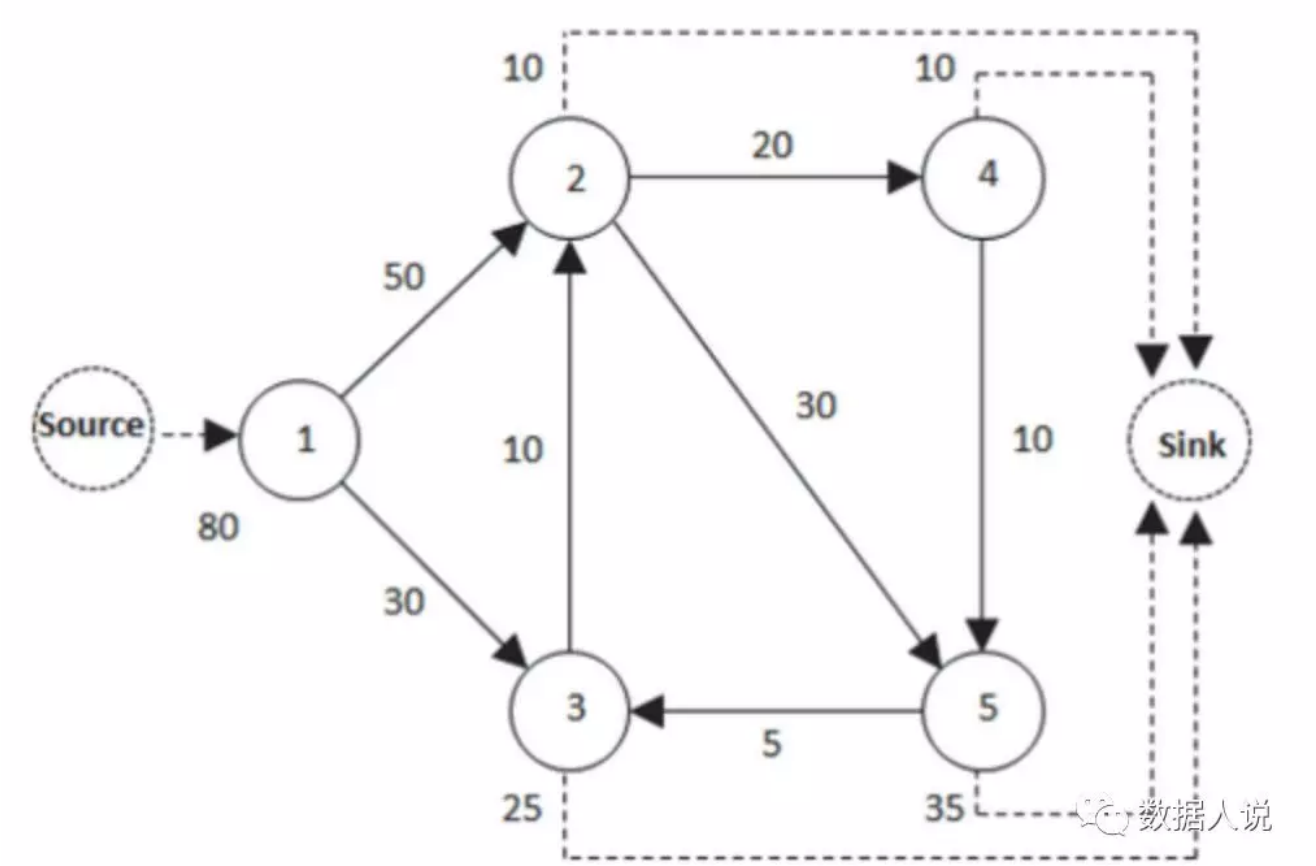
```
(py27) bash-3.2$ deepwalk --format mat --input example_graphs/blogcatalog.mat --max-memory-data-size 0 --number-
walks 80 --representation-size 128 --walk-length 40 --window-size 10 --workers 1 --output
example_graphs/blogcatalog.embeddings
Number of nodes: 10312
Number of walks: 824960
Data size (walks*length): 32998400
Data size 32998400 is larger than limit (max-memory-data-size: 0).  Dumping walks to disk.
Walking...
Counting vertex frequency...
Training...

python example_graphs/scoring.py --emb example_graphs/blogcatalog.embeddings --network example_graphs/blogcatalog.mat -
-num-shuffle 10 --all
('Results, using embeddings of dimensionality', 128)
-------------------
('Train percent:', 0.1)
('Shuffle #1:   ', {'micro': 0.36203582686245867, 'macro': 0.214281957188839})
('Shuffle #2:   ', {'micro': 0.3641818601430879, 'macro': 0.2141558113893493})
('Shuffle #3:   ', {'micro': 0.36009526736324526, 'macro': 0.20554278109497107})
('Shuffle #4:   ', {'micro': 0.35711548777684116, 'macro': 0.20457723079686158})
('Shuffle #5:   ', {'micro': 0.3657243816254417, 'macro': 0.2176258294831836})
('Shuffle #6:   ', {'micro': 0.35704441041347623, 'macro': 0.20724109677368666})
```


3. 案例一

面对实际问题，同样会转换成下面的网络结构图，网络结构和关系的含义因场景而定，但方法是一样的。

网络图：




```python
def deepwalk(_g, _corpus_num):
    _corpus = []
    for i in range(_corpus_num):
        sentence = ['o']  # 'o'为源节点 's'表示汇节点
        current_word = 'o'
        while current_word != 's':
            _node_list = []
            _weight_list = []
            for _nbr, _data in _g[current_word].items():
                _node_list.append(_nbr)
                _weight_list.append(_data['weight'])
            _ps = [float(_weight) / sum(_weight_list) for _weight in _weight_
            sel_node = roulette(_node_list, _ps)
            sentence.append(sel_node)
            current_word = sel_node
        _corpus.append(sentence)
    return _corpus

# 轮盘赌模型-按概率选择指定区域
def roulette(_datas, _ps):
```

```
    return np.random.choice(_datas, p=_ps)

# word2vec训练词向量
def word_sequence_2_vec(_word_sequence, _dim):
    model = gensim.models.Word2Vec(_word_sequence, min_count=1, size=_dim)
    global NODES
    NODES = model.vocab.keys()
    _em_vec = []
    for _node in NODES:
        _em_vec.append(model[_node].tolist())
    global NODES_NUM
    NODES_NUM = len(NODES)
    return np.mat(_em_vec)
```

```
#测试
import networkx as nx
g = nx.DiGraph()
g.add_weighted_edges_from([('o', '1', 80), ('1', '2', 50), ('1', '3', 30), ('
('2', 's', 10), ('3', '2', 10), ('3', 's', 25), ('4', '5', 10), ('4', 's', 10

dim = 3
num = 10
corpus = deepwalk(g, num)  # num个句子
embed_vec = word_sequence_2_vec(corpus, dim)

#最终的结果
'''
embed_vec =
[[-0.10715285 -0.02122051  0.05372581]
 [-0.00904658 -0.01074309  0.09090474]
 [-0.16823524 -0.16535626  0.15866642]
 [-0.14700833  0.03448284  0.15175475]
 [ 0.12771051 -0.02255488 -0.02870698]
 [ 0.02428926 -0.14465304 -0.16563286]
 [-0.00537517  0.15891171  0.00340934]]
 '''
```

3. 案例二

在电商领域类似商品主配关系图、商品同构关系网络及用户相关的关系网络等都可以用DeepWalk模型去挖掘。下面案例是基于图的结构来计算页面之间的相似度。

```
df = pd.read_csv("space_data.tsv", sep = "\t")
```

```
df.tail()
```

	source	target	depth
3323	geosynchronous satellite	tundra orbit	3
3324	geosynchronous satellite	polar mount	3
3325	geosynchronous satellite	satellite television	3
3326	deliberate crash landings on extraterrestrial ...	flyby (spaceflight)	3
3327	deliberate crash landings on extraterrestrial ...	space rendezvous	3


```
G=nx.from_pandas_edgelist(df, "source", "target", edge_attr=True, create_using
```

```
# function to generate random walk sequences of nodes
def get_randomwalk(node, path_length):
    random_walk = [node]
    for i in range(path_length-1):
        temp = list(G.neighbors(node))
        temp = list(set(temp) - set(random_walk))
        if len(temp) == 0:
            break

        random_node = random.choice(temp)  #邻接节点随机取一个
        random_walk.append(random_node)
        node = random_node
    return random_walk
```

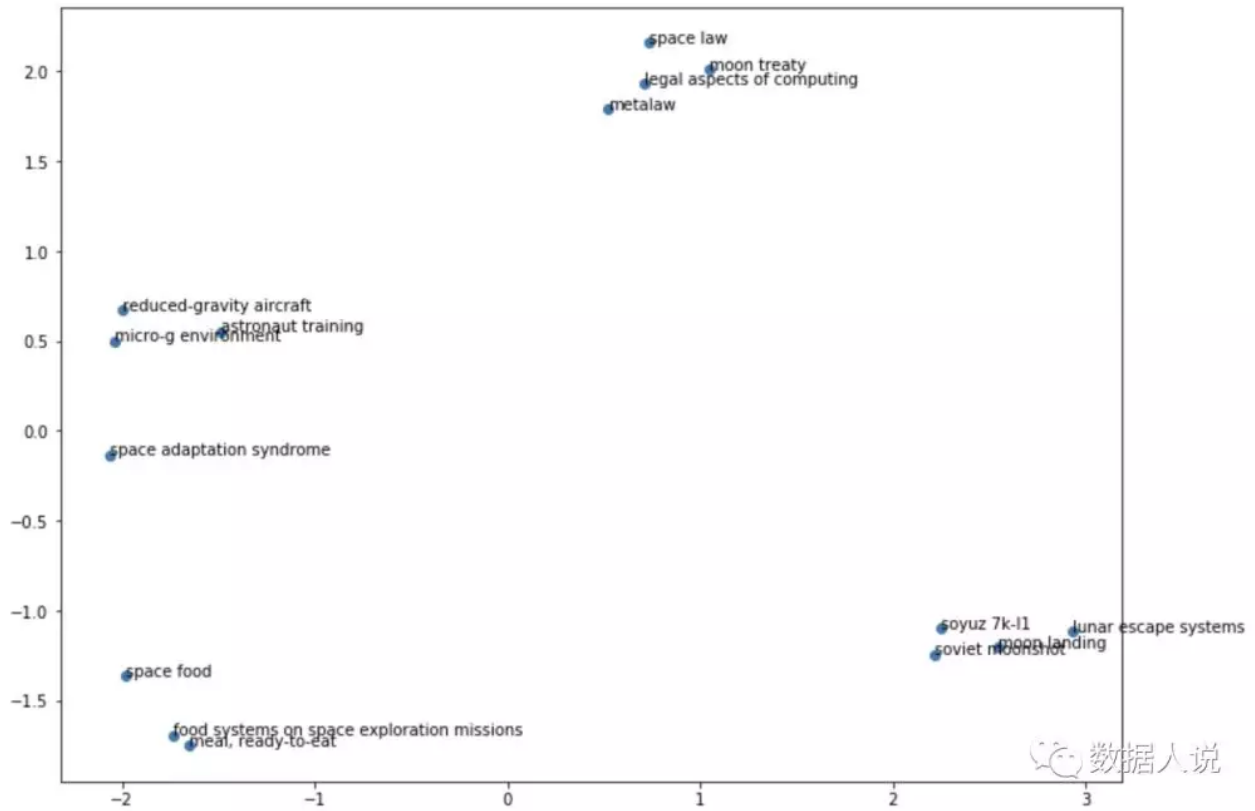
```
all_nodes = list(G.nodes())
random_walks = []
for n in tqdm(all_nodes):
    for i in range(5):  #一共更新5次
        random_walks.append(get_randomwalk(n,10))
```

```
model = Word2Vec(window = 4, sg = 1, hs = 0,
                 negative = 10, # for negative sampling
                 alpha=0.03, min_alpha=0.0007,
                 seed = 14)
model.build_vocab(random_walks, progress_per=2)
model.train(random_walks, total_examples = model.corpus_count, epochs=20, repo
```

```
#挑选部分网页看看效果
terms = ['lunar escape systems','soviet moonshot', 'soyuz 7k-l1', 'moon landi
         'space food', 'food systems on space exploration missions', 'meal, re
         'space law', 'metalaw', 'moon treaty', 'legal aspects of computing',
         'astronaut training', 'reduced-gravity aircraft', 'space adaptation s
```

最后，相似网页可视化效果

```
[27]: plot_nodes(terms)
```



4. 模型评估

通过DeepWalk能够学习得到所有顶点的连续性特征表示，然后利用这些特征，可以做聚类，异常检测，半监督分类等。评估效果应实际项目中的问题而定。

5. 总结

DeepWalk模型算是网络表示学习（Network Embedding）的开山之作，它将NLP中词向量的思想借鉴过来做网络的节点表示，提供了一种新的思路，后面Weighted DeepWalk、LINE、Node2Vec等有好几篇论文使用的也是这种思路，都是利用随机游走的特征构建概率模型，用词向量中Negative Sampling的思想解决相应问题。

最后，水平有限，如有错误，请指正，并且感谢前辈们分享的文章和经验。

参考文献：

[1] https://github.com/phanein/deepwalk
[2] https://arxiv.org/pdf/1403.6652
[3] http://www.perozzi.net/projects/deepwalk/

阅读原文