大规模数据处理技术 Assignment1

1 实验介绍

1.1 数据集介绍：Stanford web graph

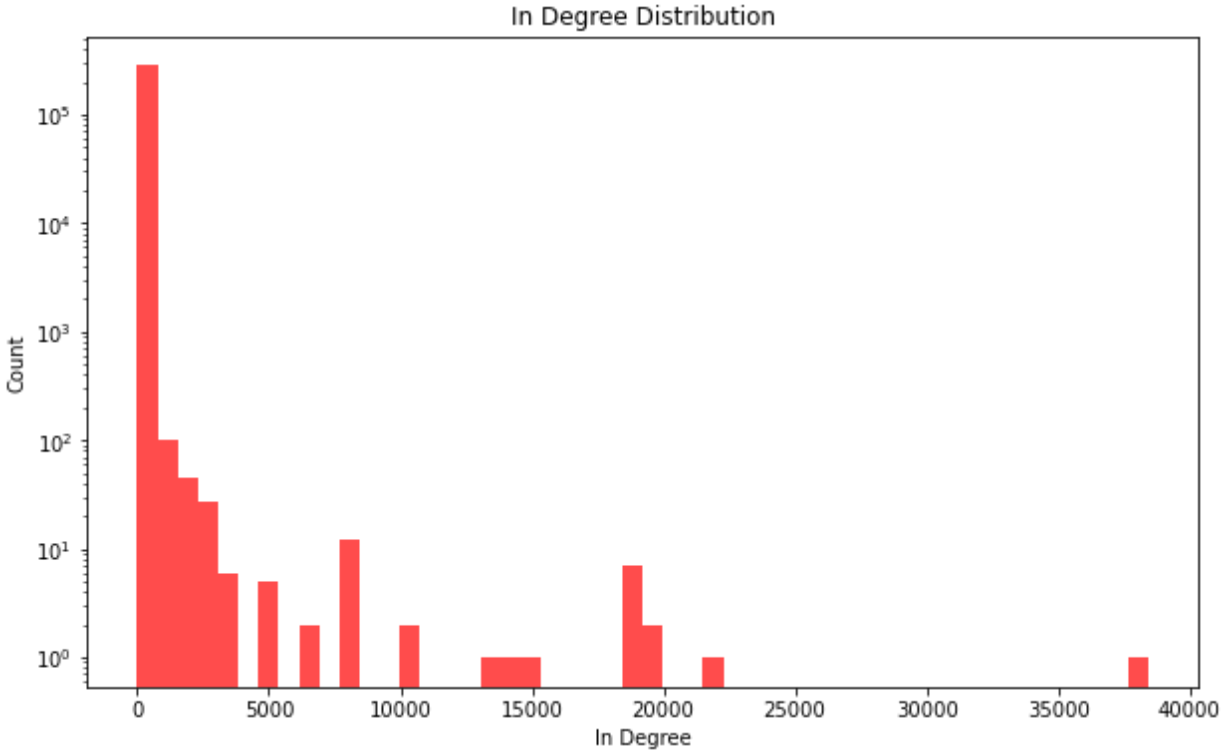
1.2 数据结构

数据集为以 ',' 分隔的 web_links.csv 表格文件。该表包含2列2,300,000行，其中第一行为表头，之后的每一行为两个整数表示一条边的两个节点。

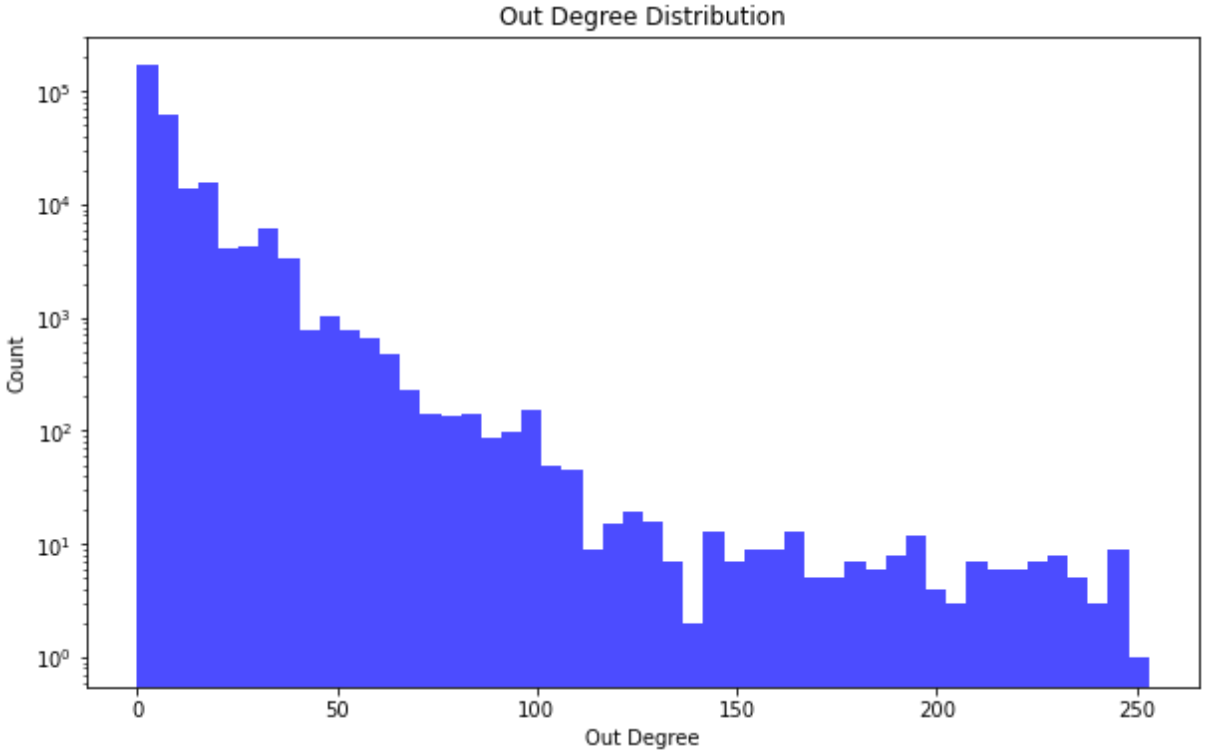
- FromNodeId：表示该边出发节点对应的Id
- ToNodeId：表示该边终止节点对应的Id

数据集中节点Id范围为1~281903，其中有281873个节点为非孤立节点，即至少有一条出边或入边。非孤立节点中，281579个节点拥有出度，261210个节点拥有入度。

所有节点入度分布直方图如下：



所有节点出度分布直方图如下：



2 算法介绍

2.1 PageRank

PageRank是Google公司创始人拉里·佩奇和谢尔盖·布林于1996年提出的一种网页重要性排序算法。PageRank的核心思想是通过网页之间的超链接关系来计算网页的重要性，即一个网页的重要性取决于指向该网页的其他网页的数量和重要性。

PageRank的计算公式如下：

$$PR(p_i) = \beta \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)} + \frac{1-\beta}{N}$$

其中，$PR(p_i)$表示网页p_i的PageRank值，$M(p_i)$表示指向网页p_i的网页集合，$L(p_j)$表示网页p_j的出度，N表示网页总数，β为阻尼系数。

3 算法实现

3.1 PageRank

```
def pagerank(edges):
    page_weights = np.ones((edges.shape[0],1))/edges.shape[0]
    beta = 0.8

    for i in range(1000):
        new_page_weights = beta * edges @ page_weights + (1-beta)/edges.shape[0]
        if np.allclose(new_page_weights, page_weights, atol=1e-12):
            break
        page_weights = new_page_weights
    return page_weights
```

上述代码实现了PageRank算法，其中 `edges` 为归一化后的邻接矩阵， `page_weights` 为每个节点的PageRank值， `beta` 为阻尼系数。此算法限制最大迭代次数为1000次，当两次迭代之间的所有PageRank值绝对值相差小于10^{-12}时，提前停止迭代。

3.2 算法优化

原数据集大约包含28万个节点，直接使用邻接矩阵大约需要消耗 584GiB 内存，而使用稀疏矩阵大约只需要 27MiB 内存。

在归一化时，为了避免numpy的广播机制隐式创建稠密矩阵，需要将系数向量转化为稀疏对角矩阵并使用矩阵乘法进行计算。

```
def normalize(mx):
    rowsum = np.array(mx.sum(1))
    rowsum[rowsum == 0] = 1
    r_inv = np.power(rowsum.astype(float), -1).flatten()
    r_mat_inv = sp.diags(r_inv)
    mx = r_mat_inv.dot(mx)
    return mx
```

4 实验结果

4.1 运行环境

- CPU: Intel(R) Xeon(R) Silver 4210R CPU
- OS: Ubuntu 20.04.6 LTS Linux 5.4.0-177-generic
- Python: Python 3.9.7
- pandas: 1.3.4
- numpy: 1.26.4

4.2 时间开销

- 邻接矩阵归一化: 119 ms ± 336 µs
- pagerank: 19 ms ± 85.3 µs