2019 春数据科学导论

PageRank 算法实现和理论作业

李梓童 2017202121

○、OS & Python

Windows

Python 3.7.1

一、PageRank

1. 代码实现思路

PageRank 代码实现总体分为两部分：

第一部分，Proc_File 函数，处理输入文件，将文本转化为有向图。在该函数中，先读取文件全部内容，作为整个字符串输入。根据换行符，将字符串切割成边数据。根据边数据中的“,”，读取边的出发点和结束点，放入 DiGraph 中，最后返回有向图。

Proc_File_2 函数为处理 Epinions 数据集设计，整体和 Proc_File 相似，只是在边数据切割过程中以“\t”为分割符。

第二部分，用循环递推计算 PageRank，并对所得结果进行排序。此处用 ranks_store 来拷贝上一轮结束时的向量结果。根据下图公式进行迭代计算，迭代轮数为 40。

在下图公式中可以看出，对于孤立点我们引入参数α进行平衡。引入α后，由于下面的算法，没有节点的 PageRank 会是 0。所以，该公式通过数学系统给了每个页面一个最小值。

$$p_i = \frac{1-\alpha}{n} + \alpha \sum_{j \to i} \frac{p_j}{m_j} = \frac{1-\alpha}{n} + \alpha \sum_{j=1}^{n} \frac{L_{ji}}{m_j} p_j$$

2. 测试数据集上运行结果

- 运行过程和结果截图

  （此处仅保留了五位小数，截图中选取了前 15 个点）

```
In [7]: runfile('C:/Users/李梓童/Desktop/tmp/Graph/
DS_lab4_pagerank/test.py', wdir='C:/Users/李梓童/Desktop/tmp/
Graph/DS_lab4_pagerank')
75879
{18: '0.00466', 737: '0.00288', 1719: '0.00215', 790: '0.00212',
118: '0.00206', 136: '0.00204', 143: '0.00200', 40: '0.00159',
1619: '0.00153', 4415: '0.00148', 1179: '0.00138', 1621:
'0.00134', 128: '0.00133', 77: '0.00127', 849: '0.00127'}
```

- 程序运行时间

5min22s

- top-10 节点及其分数

{18: '0.00466', 737: '0.00288', 1719: '0.00215', 790: '0.00212', 118: '0.00206', 136: '0.00204', 143: '0.00200', 40: '0.00159', 1619: '0.00153', 4415: '0.00148'}

二、PPR

1．代码实现思路

PPR 代码实现总体分为三部分：

第一部分，Proc_Graph_File 函数，和 PageRank 函数相似。

Proc_File_2 函数为处理 Epinions 数据集设计，整体和 Proc_File 相似，只是在边数据切割过程中以“\t”为分割符。

第二部分，Proc_Seed_File 函数，处理种子文件。方法和处理图文件相似，只是分割后的数据存到 dict 中去，dict 的 key 为字符串，value 为种子的数值。

第三部分，PPR 函数，实现和 PageRank 相似，只是在计算 ranks[key]的值时，要用(1-d)*seed（种子值）而不是(1-d)/V（求均值）。

2．测试数据集上运行结果

- 运行过程和结果截图

```
In [73]: runfile('C:/Users/李梓童/Desktop/tmp/Graph/PPR.py',
wdir='C:/Users/李梓童/Desktop/tmp/Graph')
75879
[('18', 0.007074413678653963), ('31', 0.00602489193445312),
('27', 0.00566967539286199), ('40', 0.005625375823553982),
('34', 0.00553319280644918), ('30', 0.0055232008718673915),
('0', 0.0053466198912624635), ('1', 0.005099990519832233),
('12', 0.005037850125698701), ('28', 0.004832582656223906),
```

- 程序运行时间

6min9s

- top-10 节点及其分数

[('18', 0.007074413678653963), ('31', 0.00602489193445312), ('27', 0.00566967539286199), ('40', 0.005625375823553982), ('34', 0.00553319280644918), ('30', 0.0055232008718673915), ('0', 0.0053466198912624635), ('1', 0.005099990519832233), ('12', 0.005037850125698701), ('28', 0.004832582656223906)]

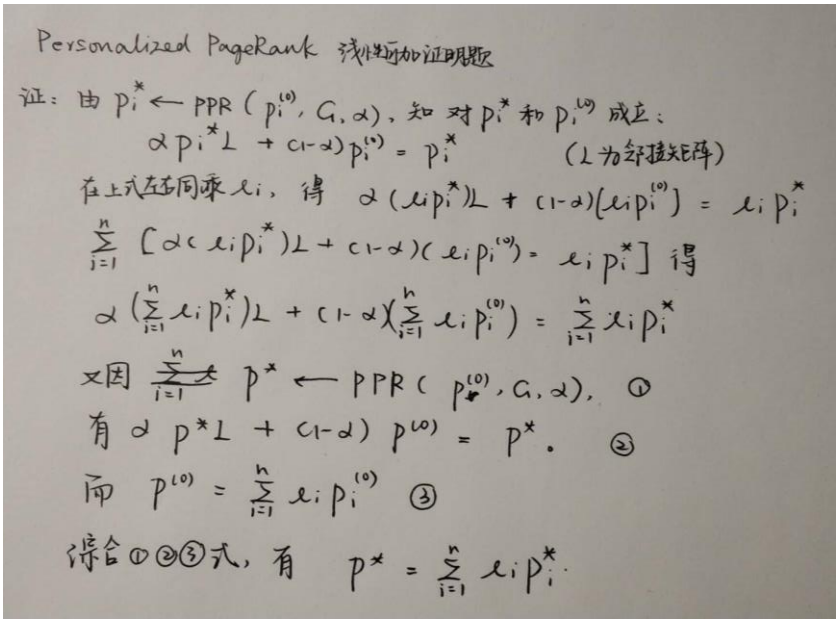
三、如何应对大规模图数据的设想和实现

从存储方式上看，大规模图数据可以用稀疏矩阵、分布式数据库进行存储。稀疏矩阵存储能节省的空间毕竟是有限的，分布式存储则从根本上改变了存储方式。

从计算处理上看，大规模图数据可以通过运用 MapReduce 模型、BSP 计算框架及其分

解框架 GAS 模型等手段提高计算效率。老师上课时提到一点 Multi-GPU 可以大大加快计算速度的例子，但只是一笔带过，因为时间原因没有细说。分布式处理应该是未来大规模图数据处理的趋势，通过并发来加快处理速度。

四、Personalized PageRank 线性可加证明题



五、实验小结

个人认为 PageRank 算法的中心思想是将一个级别/重要性的排序问题转化成了一个以群体民主投票的方式求解的问题，图节点之间的邻接关系即被认为是投票行为。同时，由于各个节点的权重（重要性）不同，重要的节点投票具有较大的分量，PageRank 值避免了几何计算中心度等方法的偏差，较完整地体现了重要性。