PageRank 算法作业报告

陈小康 1500012741

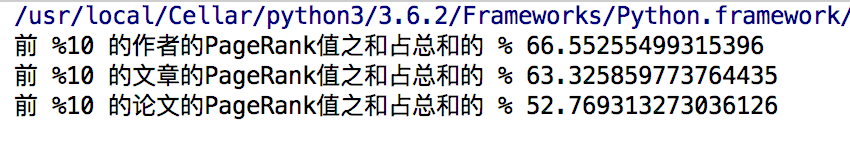
1. 实现环境

* 操作系统为mac OS sierra 10.12.6
* 使用的编程语言为Python 3.6.2
* 调用的Python库为：pygraph（下载到本地），re

1. 涉及的数据
2. 计算用到的数据
   1. 计算author的PageRank值

* aan/release/2014/networks/author-citation-network-nonself.txt
  1. 计算article的PageRank值
     + aan/release/2014/networks/paper-citation-network-nonself.txt
     + aan/release/2014/acl-metadata.txt
  2. 计算venue的PageRank值
     + aan/release/2014/acl-metadata.txt
     + aan/release/2014/networks/paper-citation-network-nonself.txt

1. 输出的数据
   1. author的PageRank值——author\_rank.txt
   2. article的PageRank值——article\_rank.txt
   3. venue的PageRank值——venue\_rank.txt
2. 具体实现方式
3. 首先是构造引用关系图。这里引用第三方库”pygraph”，里面包含了add\_node(), add\_edge(), has\_node(), has\_edge(), neighbors()等内置函数，可以很方便的建图。
   1. 对于author的信息，直接用引用关系网，构造author节点之间的引用关系。
   2. 对于article信息，首先用论文编号之间的引用关系建图，构建引用关系网，然后根据acl-metadata.txt，将article的编号和题目对应起来，最终输出文件时输出论文标题和对应ID的PageRank值。
   3. 对于venue信息，我的思路是利用article之间的引用关系来决定venue之间的引用关系。具体思路是这样：从acl-metadata.txt中得到每篇article投的若干venue，然后从aan/release/2014/networks/paper-citation-network-nonself.txt中获得article之间的引用关系，利用这个引用关系决定对应venue之间的引用关系。例如，article A对B有引用，A投的venue为{v1,v2,v3…vm},B投的venue为{V1,V2,V3…Vn}，那么认为v1有对V1,V2…Vn的引用，v2有对V1,V2,,,,Vn的引用，以此类推，构建venue的引用图。
4. 然后是算法的实现部分。对于已经构建好的引用图中的每个节点，赋初值 1/n（n为节点数）。然后每次迭代时，对于节点x，找出它的入点node1,node2..nodet，利用这些入点的现有PageRank值除以这些入点的出度，在相加求和，再加上随机游走算法的参数，得到本轮迭代算出的节点x的新PageRank值，但目前还不更新。等本轮迭代中所有节点均已计算过之后，统一更新PageRank值，并与前一轮的PageRank值做差、求和，若变化程度小于delta（代码中给出），则停止迭代。用for循环而不用矩阵是因为数据量过大，是一个稀疏矩阵，故直接计算效率可能更高一些。
5. 数据分析

为了考察实现的PageRank算法的效果，对于每个result文件，取前10%的数据的PageRank值相加，计算在所有元素PageRank值之和中占的比例。实验结果表明，前10%的数据占比很大，效果较为理想。（见下图）

1. 思考

在完成作业过程中，遇到几个问题，如下：

1. 元数据文件格式问题。可能是Mac系统转换的原因吧，文件到手的时候不是utf-8编码，读取的时候出现了些错误，手动用sublime text 3改了文件格式。
2. acl-metadata文件中，有一个article的title中有一个换行符，占了两行，不符合整体规范。（手动删除了这个换行符）
3. acl-metadata文件中，有两个title的结尾多了一个制表符’\t’，导致用正则表达式获取花括号内内容时会多获取一个制表符。这样在输出article名字时会多输出一个制表符。如下图