

华中科技大学

课程实验报告

课程名称： 大数据分析

专业班级： 物联网工程 1801 班
学 号： U201814500
姓 名： 王英嘉
指导教师： 崔金华
报告日期： 2020.12.9

计算机科学与技术学院

目录

| | |
|---------------------------------|---|
| 实验二 PageRank 算法及其实现..... | 1 |
| 2.1 实验目的..... | 1 |
| 2.2 实验内容..... | 1 |
| 2.3 实验过程..... | 1 |
| 2.3.1 编程思路..... | 1 |
| 2.3.2 遇到的问题及解决方式..... | 3 |
| 2.3.3 实验测试与结果分析..... | 3 |
| 2.4 实验总结..... | 4 |

实验二 PageRank 算法及其实现

2.1 实验目的

- 1、学习 pagerank 算法并熟悉其推导过程；
- 2、实现 pagerank 算法¹；（可选进阶版）理解阻尼系数¹的作用；
- 3、将 pagerank 算法运用于实际，并对结果进行分析。

2.2 实验内容

提供的数据集包含邮件内容（emails.csv），人名与 id 映射（persons.csv），别名信息（aliases.csv），emails 文件中只考虑 MetadataTo 和 MetadataFrom 两列，分别表示收件人和寄件人姓名，但这些姓名包含许多别名，思考如何对邮件中人名进行统一并映射到唯一 id？（提供预处理代码 preprocess.py 以供参考）。

完成这些后，即可由寄件人和收件人为节点构造有向图，不考虑重复边，编写 pagerank 算法的代码，根据每个节点的入度计算其 pagerank 值，迭代直到误差小于 10^{-8}

实验进阶版考虑加入 teleport β ，用以对概率转移矩阵进行修正，解决 dead ends 和 spider trap 的问题。

输出人名 id 及其对应的 pagerank 值。

2.3 实验过程

2.3.1 编程思路

预处理上 preprocess.py 实现了将原数据转化为以发件人 id 和接收人 id 两列的 dataframe 表格，并解决了其中的别名转化问题，对于名字未知 id 的邮件则忽略掉，生成的文件 sent_receive.csv 如图 1 所示，本实验在此基础上进行。

¹ 基本 pagerank 公式 $r=Mr$

```
sent_receive.csv ×
lab2 > source > sent_receive.csv
1 ,sent_id,receive_id
2 0,87,80
3 1,87,80
4 2,32,80
5 3,32,80
6 4,80,81
7 5,80,185
8 6,32,80
9 7,80,81
10 8,87,80
11 9,87,80
12 10,87,80
13 11,87,80
```

图 1

算法流程如下：

在读取数据之后，提取有向边（不考虑重复边）并按入结点 id 排序，之后通过 python 中的 set 去重获得结点个数作为转移矩阵 M 的边长，对于转移矩阵 M 中的初始值，通过查找以某结点为入结点有多少条有向边，以此更新概率，举例来说，以结点 1 为入结点有 3 条有向边，分别为(1,2),(1,3),(1,4)，那么可以参考图 2 中第一列所示。

$$M = \begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

图 2

之后还需要初始化向量 w，每个元素值为 $1.0/\text{node_num}$ ，node_num 为结点个数，以 node_num=4 为例，如图 3 所示。

$$w_0 = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}$$

图 3

阻尼系数根据要求设为 0.85，用以修正转移矩阵，之后不断迭代更新即可，当新的向量 w 与原来的 w 误差在 $1e-8$ 以内时，迭代结束，最后将结点的 pagerank

向量 w 除以 $\text{np.sum}(w)$ 后输出成 `pagerank.csv` 文件保存。

除此之外，可以尝试 `python` 中的 `networkx` 库，可以很方便地进行 `pagerank` 值的计算，以及作图直观展示等等，代码见 `pagerank(networkx).py`，生成结果见 `pagerank(networkx).csv`。

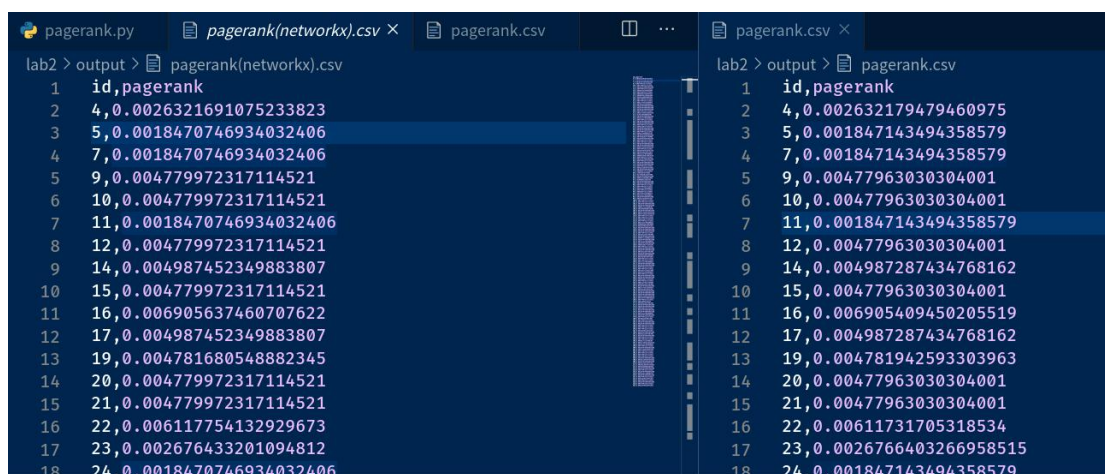
2.3.2 遇到的问题及解决方式

在算法策略上，经助教指导，也可以采用如下的流程进行：对于每条表格中的有向边，遍历并对 M 中对应位置的值计数加 1，之后对每一列求和整体做除法，但是这种策略也需要遍历一次全部有向边，因为需要知道结点个数 `node_num` 才能初始化转移矩阵 M 的维度。

一开始算出的 `pagerank` 向量的和不为 1，以为自己的算法出现了错误，后来经过仔细排查后仍然没发现，上网查阅资料后发现如果存在黑洞结点，最后的和确实可能小于 1 的，这里为了和标准结果对比，将结果除以 $\text{np.sum}(w)$ 作为最终结果。

2.3.3 实验测试与结果分析

对比自己生成的 `pagerank.csv` 和 `networkx` 库生成的 `pagerank.csv` 如图 4 所示，差异很小，说明实现的算法基本正确。



| id | pagerank |
|----|-----------------------|
| 4 | 0.0026321691075233823 |
| 5 | 0.0018470746934032406 |
| 7 | 0.0018470746934032406 |
| 9 | 0.004779972317114521 |
| 10 | 0.004779972317114521 |
| 11 | 0.0018470746934032406 |
| 12 | 0.004779972317114521 |
| 14 | 0.004987452349883807 |
| 15 | 0.004779972317114521 |
| 16 | 0.006905637460707622 |
| 17 | 0.004987452349883807 |
| 19 | 0.004781680548882345 |
| 20 | 0.004779972317114521 |
| 21 | 0.004779972317114521 |
| 22 | 0.006117754132929673 |
| 23 | 0.0026766433201094812 |
| 24 | 0.0018470746934032406 |

图 4

2.4 实验总结

通过本次实验，一方面掌握了 pagerank 的算法原理，以及阻尼系数在 pagerank 中的应用，另一方面也更熟练了用 numpy 对矩阵进行操作。除了实验中所说的阻尼系数，还额外了解了 spam 攻击防御等等，收获很大。

ⁱ 进阶版 pagerank 公式: $\mathbf{r} = \beta \mathbf{M}\mathbf{r} + (1 - \beta) \left[\frac{1}{N} \right]_{N \times 1}$, 其中 β 为阻尼系数, 常见值为 0.85