

**课 程 实 验 报 告**

**课程名称： 大数据分析**

**专业班级： 大数据2101班**

**学 号： U202115652**

**姓 名： 李嘉鹏**

**指导教师： 王蔚**

**报告日期： 2023年5月11日**

**计算机科学与技术学院**

**目录**

[实验二 PageRank算法及其实现 1](#_Toc135394315)

[**2.1实验目的** 1](#_Toc135394316)

[**2.2 实验内容** 1](#_Toc135394317)

[**2.3 实验过程** 1](#_Toc135394318)

[2.3.1 编程思路 1](#_Toc135394319)

[2.3.2 数据结构与存储形式 3](#_Toc135394320)

[2.3.3 遇到的问题及解决方式 3](#_Toc135394321)

[2.3.4 实验测试与结果分析 4](#_Toc135394322)

[**2.4 实验总结** 5](#_Toc135394323)

[**2.5 参考文献** 6](#_Toc135394324)

# 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，理解阻尼系数的作用；

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以及处理后的sent\_receive.csv数据以供参考）。

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

本实验主要是对Pagerank算法的具体实现。压缩包里已经给定了preprocess.py文件，可以直接读入别名和人名id信息，并对邮件中出现的人名进行统一。接下来，根据邮件中的发送和接收方构造有向图的边。

在核心的pagerank算法中，首先遍历收件人和发件人，找到图中全部不重复的结点。接下来构建一个邻接矩阵M并对其初始化，使其在每一对（发件人，收件人）的位置上的值都为1，其余位置上的值为0，其边长为所有结点的总数N。

此后，对M的每一列进行标准化，具体操作是使每一列的和等于1。实现标准化的代码如下。

|  |
| --- |
| **for** j **in** range(N):  sum\_of\_col = sum(M[:, j])*#取得每一列的矩阵之和* **for** i **in** range(N):  **if** M[i, j]:  M[i, j] /= sum\_of\_col |

做好上述准备工作后，需要实现算法的核心模块——迭代。定义误差e、迭代次数k、阻尼系数β，并分别将其初始化为1、0、0.85（阻尼系数β的取值一般在0.8到0.9之间，后续会进行调参并观察迭代次数随β的变化）。每次迭代中，用next\_r记录新的r矩阵，根据公式可得到next\_r的值。注意在任何一次迭代中，都要再次将r标准化。若某时刻计算得到的误差e小于10-8，则停止迭代。迭代部分的代码如下。

|  |
| --- |
| **while** e > 0.00000001: *# 迭代到误差小于10e-8停止* next\_r = np.dot(M, r) \* b + (1-b) / N \* np.ones(N)sum\_of\_col = sum(next\_r)  next\_r = next\_r / sum\_of\_col *#确保整个矩阵值的和保持为1* e = next\_r - r  e = max(map(abs, e)) *# 计算误差* r = next\_r  k += 1 |

最后要输出人名id及其对应的pagerank值，直接采用数组循环输出即可。

算法的整体流程图如图2-1所示。

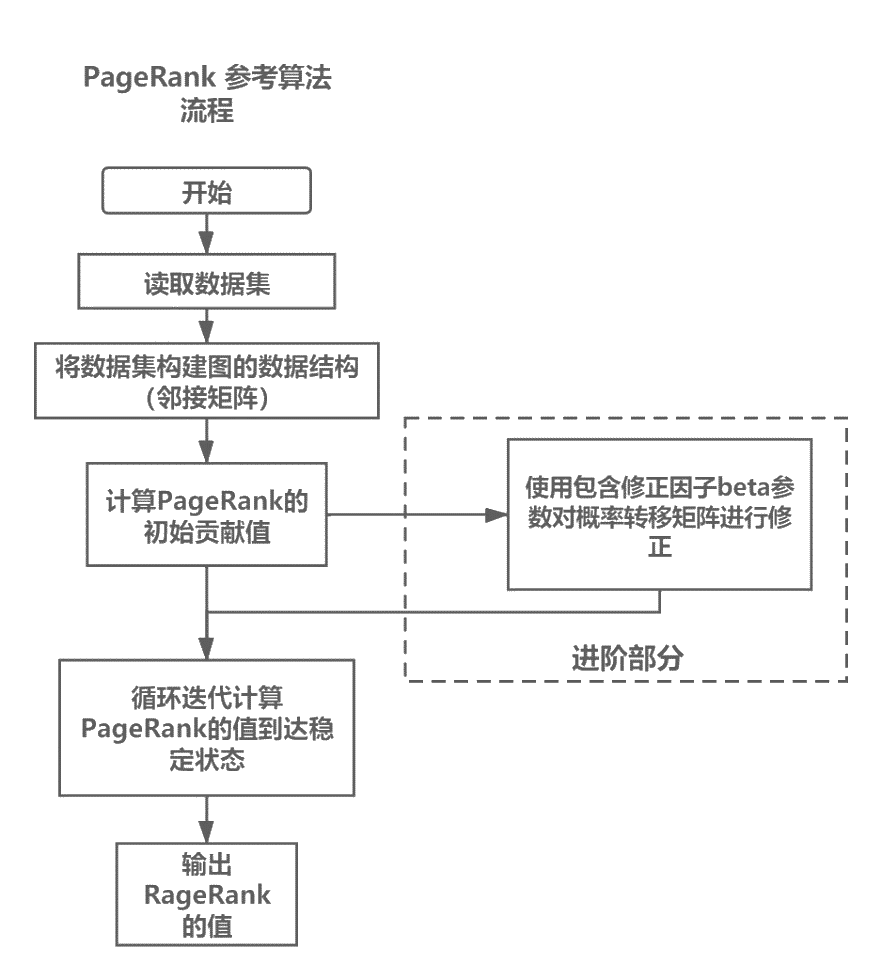


图2-1：算法流程图

### 2.3.2 数据结构与存储形式

**（1）edges**

①类型：二维数组

②作用：存储每条边的起点和终点（即发件人和收件人）。

**（2）nodes**

①类型：一维数组

②作用：存储当前所有的结点（发件人和收件人）。

**（3）M**

①类型：大小为N\*N的矩阵（N为nodes的大小）

②作用：邻接矩阵，记录每一对用户之间的(sent, receive)，若发件人a给收件人b发邮件，则M[a][b] = 1，否则为0。

**（4）next\_r**

①类型：大小为1\*N的矩阵，即一维数组

②作用：存储每次迭代后所有人的pagerank值。

### 2.3.3 遇到的问题及解决方式

实验中主要遇到了两个问题，其一是Python解释器没有配置好，导致一些库函数调用的时候报错，实际上只需要在Project Settings中设置好相应的环境即可解决。

其二是在迭代的过程中，忘记将r标准化了，导致每次r中元素的总和都不等于1，最后收敛时的pagerank总和也不等于1。这里当时调bug找了好半天，实际上只需要进行标准化就能解决，经验证，最后输出的pagerank结果的和为1，完美解决了这一问题。

### 2.3.4 实验测试与结果分析

在β=0.85下运行pagerank.py，输出的人名id与pagerank值如图2-2所示（节选一部分）。可以看到总迭代次数为55次。

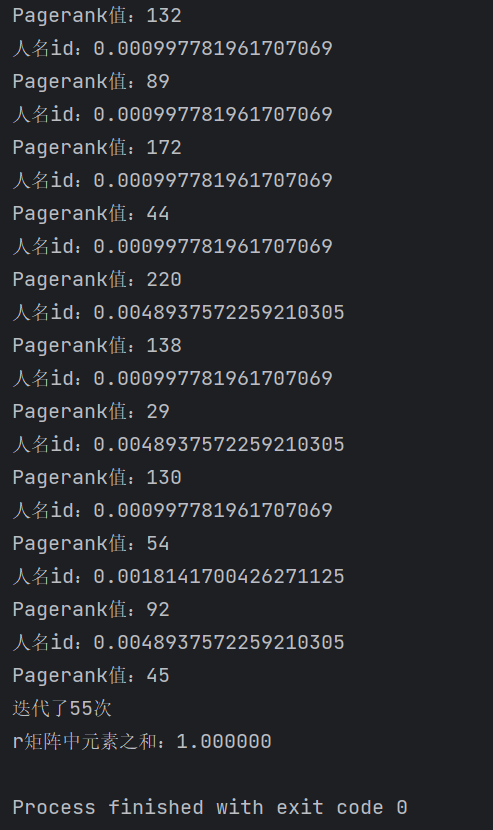
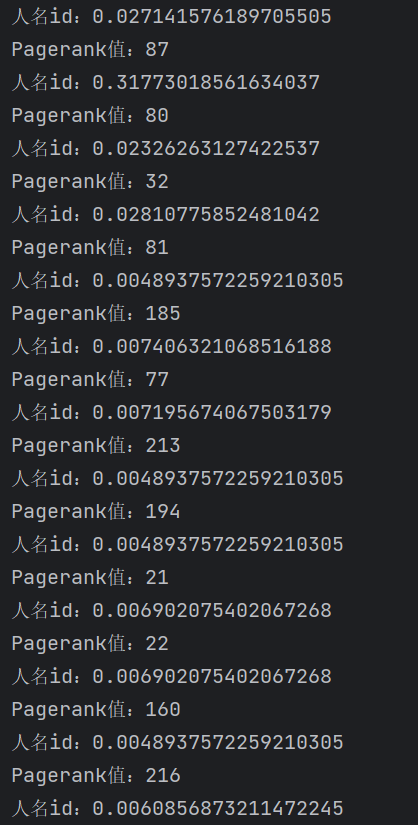


图2-2：β=0.85时的人名id、pagerank值与迭代次数

为了进一步观察所有人的Pagerank值，直接输出整个r矩阵，如图2-3所示。

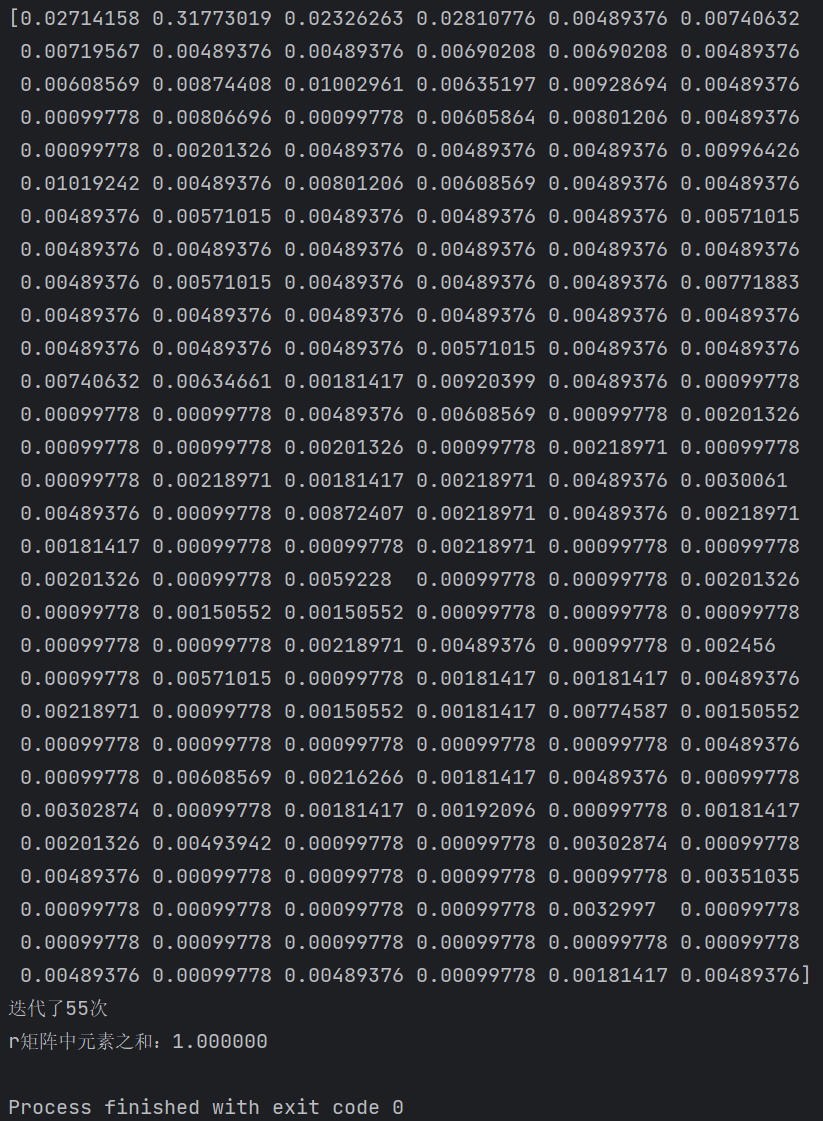
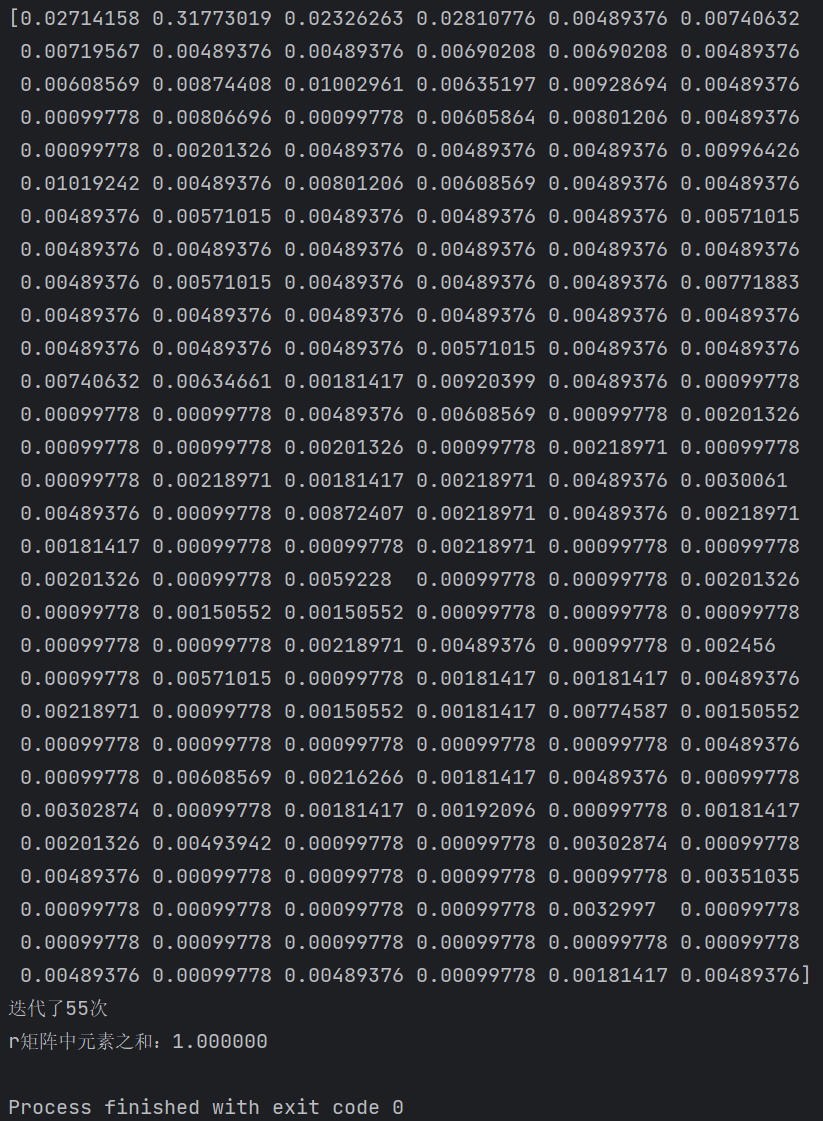


图2-3：记录Pagerank值的r矩阵的最终状态

分别改变β=0.8和β=0.9，得到不同的迭代次数分别为46次和69次，如图2-4、图2-5所示。

与β=0.85时迭代55次相比较，上述现象出现的原因是β越大，每次迭代中teleport的概率越低，越容易陷入spider trap和dead ends的问题，导致收敛越慢。若取β=1，则Pagerank算法回归到最原始的版本，如果图中存在spider trap和dead ends，则算法有可能无法收敛。这也很好的解释了为什么目前网页重要度排序中一般都会用β来调节teleport的效果。

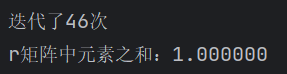
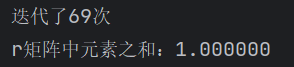
 

图2-4：β=0.8时的迭代次数 图2-5：β=0.9时的迭代次数

## **2.4 实验总结**

本次实验中，我具体实现了Pagerank算法，进一步理解了一个重要的互联网网页重要度排序方法。

它的亮点主要是在有向图上定义一个随机游走模型，通过一定概率跳出，避免了死循环等问题。通过这种算法，可以很快速地对网页等级进行标识，同时还能应用于社会影响力分析、文本摘要等多个领域。

本次实验还进一步让我掌握了利用Python实现迭代算法的能力，在代码熟练度上有很大提升。

## **2.5 参考文献**

[1] 图机器学习——3.2 PageRank 算法问题改进与延伸 <https://blog.csdn.net/weixin_41929524/article/details/121736652>

[2] cs224w（图机器学习）2021冬季课程学习笔记4 Link Analysis: PageRank (Graph as Matrix) <https://developer.aliyun.com/article/1053533>