

**课 程 实 验 报 告**

**课程名称： 大数据分析**

**专业班级： 校交1801**

**学 号： U201814132**

**姓 名： 张浚哲**

**指导教师： 王蔚**

**报告日期： 2020年12月30日**

**计算机科学与技术学院**

**目录**

[实验二 PageRank算法及其实现 1](#_Toc57042479)

[**1.1实验目的** 1](#_Toc57042480)

[**1.2 实验内容** 1](#_Toc57042481)

[**1.3 实验过程** 1](#_Toc57042482)

[1.3.1 编程思路 1](#_Toc57042483)

[1.3.2 遇到的问题及解决方式 1](#_Toc57042484)

[1.3.3 实验测试与结果分析 1](#_Toc57042485)

[**1.4 实验总结** 2](#_Toc57042486)

# 实验二 PageRank算法及其实现

## **1.1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。

## **1.2 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。

## **1.3 实验过程**

### 1.3.1 编程思路

本次实验主要需要编写的函数有：

1. 结点类
2. 结点数据统计
3. 转移矩阵生成函数
4. 迭代函数
5. 结点类（class node）

初始一个结点类的结点需要传入id，链出的目的结点列表des\_nodes，链入的源结点列表source\_nodes，以及结点本身的rank。

其本身初始化时，还有两个属性，一个是链出的目的结点列表的个数，一个是链入的源结点列表的个数。其初值都为0.

结点类还包括一个添加目的结点的函数和一个添加源结点的函数。分别为adddes和addsource函数。均是将对应结点的序号添加到对应的结点列表中，并将对应结点总数加1.

1. 结点数据统计

打开数据集sent\_receive.csv，将其中每一行读取出来。在遍历数据集的过程中，做一个结点字典，将每一条信息的源结点和目的结点的键对应的值设为1，遍历完后统计字典的长度，便可以知道本数据集中真实有效的结点数目的个数了。结点的编号最高为462，最低位1，但是目的结点加上源结点一共只出现了180个不同的结点，因此在初始化462个结点时，每个结点的rank只需要设置成1/180即可。然后在之前遍历数据的同时，对每一条数据调用adddes和addsource操作，这样便可以把每个结点的信息加载到对应的462个结点实例中。

1. 转移矩阵生成函数（matrix\_make）

入口参数：allnodes（所有的结点实例组成的列表）。

返回值：matrix（生成的转移矩阵）。

功能：根据所有的结点信息，生成此结点集的转移矩阵。

实现：首先得到结点类中结点的个数，生成对应个数-1的全0方阵。然后遍历结点集中的每个结点，对其中的一个结点，遍历其所有的源结点，由于转移矩阵第一行乘原来结点的rank产生的是新的第一个结点的rank，所以转移矩阵第一行对应的值就是对应元素对第一个结点的链入的权值。而只有在结点的源结点列表中的结点来链入到此结点，因此转移矩阵只有对应源节点列表中的结点对应的列上才有值，注意文件中的结点是从1开始编号的，因此此处需要将对应的结点进行-1操作之后才是对应转移矩阵中的位置。具体的值又有对应源结点本身的rank和其本身的链出个数决定。即为其本身的rank除以本身的链出个数。如此一来，便能生成转移矩阵中每一个行每一列对应的值了。

1. 迭代函数（diedai）

入口参数：matrix（转移矩阵），node\_array（结点集的rank值）。

返回值：然后收敛后的new\_node\_array（结点集的新rank值）。

功能：根据得到的matrix矩阵和结点的rank值计算新的结点的rank值，直至收敛。

实现：首先将转移矩阵和结点rank值向量相乘，得到第一阶段的rank值，然后使用teleport β公式，，此处的β设置为0.85.这样算出来的结点集的新rank值才是这一次最终的rank值，然后使用这个和上一次的node值做差进行计算，取其sse，当相邻的两次sse小于10^-8时停止迭代，返回结点集最新的rank值。

### 1.3.2 遇到的问题及解决方式

1.一开始得到的rank值非常的小，对应的结果也不正常，后来发现是没有归一化造成的，由于dead ends的存在会导致整个结点集中的rank值不断地减小，最后的总值之和不为1.就会导致一些计算上的问题。因此，在每次完成最终阶段的计算后，都要进行一次归一化这里需将结点集中的rank提取出来，如果直接使用numpy的sum函数最后进行归一化，得到的结果不完全是1.归一化之后的结果和得分就会好的多。

2.在未引入teleport β时，正如之前所分析的，462个结点只出现了180个结点，所以此结点集中的dead ends非常多，很容易就造成了提前收敛，这就是为什么不加teleport β，整个碟台过程只执行了5次就收敛了，且结果还不是很准确。但是在加如teleport β之后，需要迭代30多次才能收敛，且结果趋于正常。

### 1.3.3 实验测试与结果分析

1.首先统计sent\_receive.csv中结点的信息，具体统计方法可见编程思路中的结点数据统计，统计出来的结点数据如图1.1所示：

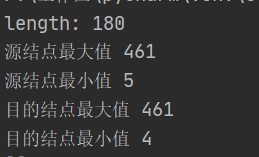


图1.1

可见，真正出现的结点种数只有180个，但是结点的最大编号却为461，为了方便，我们设置462个大小的结点集，并将结点的编号作为在结点集中的下标。

产生的转移矩阵如图1.2所示：

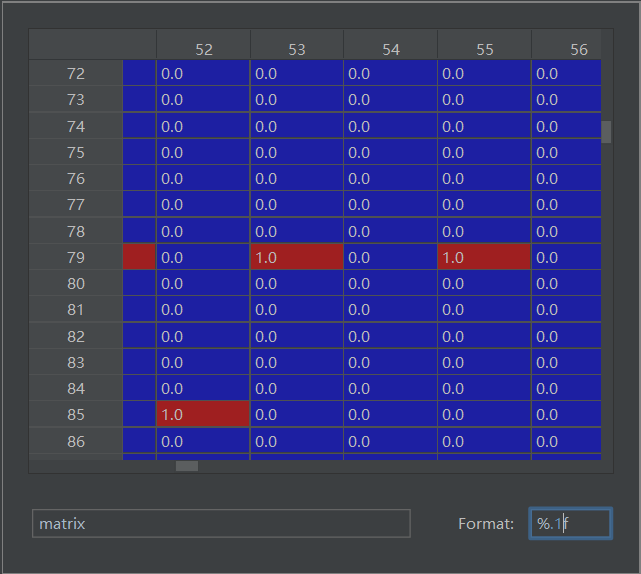


图1.2

可见，此转移矩阵比较稀疏，容易出现dead ends的问题。

最终的迭代过程如图1.3所示：

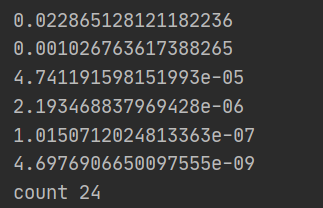


图1.3

一共执行了24次收敛，上图的loss是每4次迭代计算的一次loss。

最后得到的结点按rank从大到小排序的结果如图1.4所示：

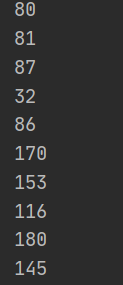


图1.4

这里选了rank前10的人的id。

## **1.4 实验总结**

pagerank算法本身不难，主要是对应结点数据的获得和对应转移矩阵的生成，注意pagerank本身的算法，只有链入到对应结点的结点才会对对应结点进行权值的转移，而转移权值的多少又取决于对应结点的出度，而且此数据集中结点编号最大为462，但是一共只出现了180个不同的编号，此数据集中一共有7533条数据，整个图还是比较稀疏的图，因此存在很多dead ends的问题，同时，由于只有几个人收发的邮件比较多，因此很容易就会组成环，因此也会产生很多spider trap的问题，所以需要使用teleport β来避免对应的rank值的流失。

整个实验做下来，对pagerank的理解加深了许多，只上课虽然也能听明白pagerank大概是什么，但是只有复现过一次才知道其中许多的细节问题。比如矩阵怎样生成，结点的rank怎样迭代等。对对应的算法不熟悉的话应该结合课上的ppt和网上的博客，只有先把算法本身的过程看懂才能保证实现的算法不会在逻辑上出问题。