

**课 程 实 验 报 告**

**课程名称： 大数据分析**

**专业班级：**

**学 号：**

**姓 名：**

**指导教师：**

**报告日期：**

**计算机科学与技术学院**

**目录**

[实验二 PageRank算法及其实现 1](#_Toc134644946)

[**1.1实验目的** 1](#_Toc134644947)

[**1.2 实验内容** 1](#_Toc134644948)

[**1.3 实验过程** 1](#_Toc134644949)

[1.3.1 系统整体框架 1](#_Toc134644950)

[1.3.2 编程思路与实现 2](#_Toc134644951)

[1.3.3 实验测试 4](#_Toc134644952)

[1.3.3 对比实验及结果分析 4](#_Toc134644953)

[**1.4 实验总结** 4](#_Toc134644954)

# 实验二 PageRank算法及其实现

## **1.1实验目的**

1、学习pagerank算法，了解其计算过程，并熟悉其推导过程；

2、实现pagerank算法，理解阻尼系数的作用，运用阻尼系数优化pagerank算法，减少算法本身的漏洞；

3、将pagerank算法运用于实际，实现pagerank算法以及加入teleport β后的修正过程，对结果进行对比分析。

## **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。

## **1.3 实验过程**

### 1.3.1 系统整体框架

PageRank算法最开始是用来评估网页重要性的算法，是LinkAnylsis领域的重要算法。PageRank算法基于一个简单的假设，即一个页面的重要性取决于其他页面对它的引用。如果一个页面被其他页面引用得越多，那么它的权重就越高。同时，如果一个页面的权重很高，那么它引用的其他页面的权重也会相应地提高。[1]。在基础的PageRank算法上，学者引入了teleport β，用来避免dead ends，spdier trap等问题。图2-1非常形象的展示了PageRank的基本假设。[2]图2-2则简单展示了在原有算法基础上加入teleport β机制的效果。[3]

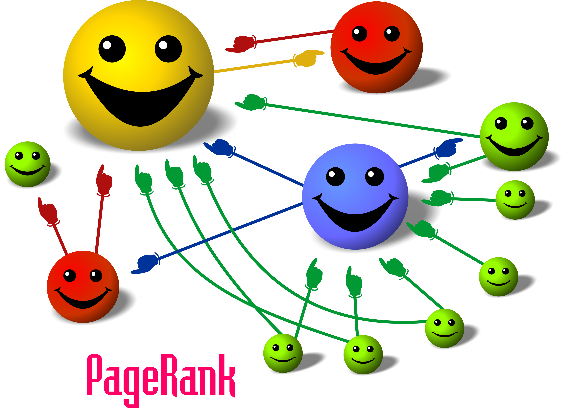


图2-1 PageRank算法概念图

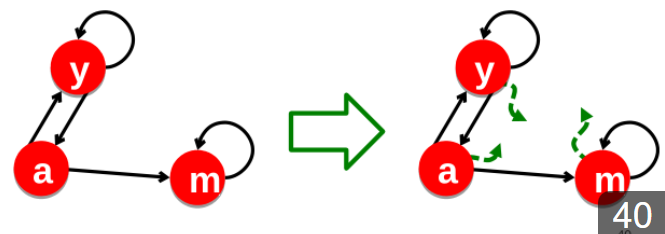


图2-2 PageRank算法加入teleport β机制示意图

### 1.3.2 编程思路与实现

**1）数据处理**

由于数据已经被预处理过，我们得到了开箱即用的数据集send\_receive.csv。（感谢学长姐）图2-3展示了数据集的部分数据。send\_id表示发送方，receive\_id表示接收方。整个数据处理过程分为两部分，第一部分为读入信息，统一人名；第二部分为构造有向图并写入文件（图2-4展示了此部分的代码）。通过这样的处理，能够极大的方便后续矩阵的构建。

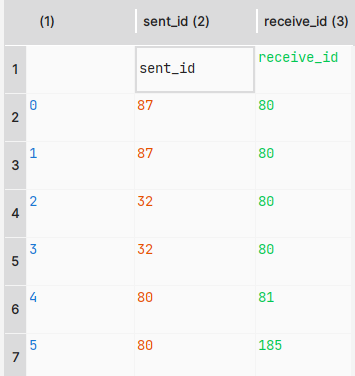
  
图2-3 send\_receive.csv数据截图



图2-4 数据预处理部分代码

**2）矩阵构建**

在构建矩阵时，根据人数，我们可以构建一个N×N的矩阵mat，其中N为总人数，mat[i][j]表示页面i是否链接到页面j，若不为0意味着界面i有链接到页面j，否则没有。之后对整个矩阵进行归一化处理，这样可以使得每个页面的PageRank值更加准确地反映其在整个网络中的重要性，避免了权重被平均分配的问题，从而提高了PageRank算法的准确性。



图2-5 矩阵构建代码展示

**3）PageRank算法及其迭代过程**

对于基本的PageRank算法，其迭代过程较为简单，数学表达式如图2-6所示。通过数学推导（此处省略具体过程），可以表示为两个矩阵相乘，一个矩阵为有向图的邻接矩阵，另一个为存储着PageRank值的pr矩阵。

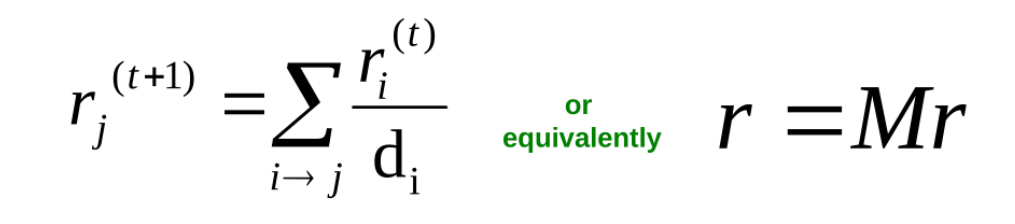


图2-7 pagerank算法计算过程

具体地，pr矩阵在初始化后进入迭代，通过和前一次计算误差err，然后更新pr矩阵并进行下一次迭代。其中err代表pagerank值的变化程度，err越小，代表pagerank值越来越趋于稳定。当误差err小于精度eps时，近似认为pagerank值可看作不变，迭代结束。



图2-7 pagerank算法迭代代码展示

**4）引入了teleport β的PageRank算法及其迭代过程**

传统的PageRank算法会遇到诸如deads end，spider trap等问题。图2-8是上述两问题的示意图，deads end意味着某一个节点没有链接到任何一个其他页面；spider trap意味着某几个节点互相链接，并且没有链接到任何一个外部页面。他们都会导致pagerank值的异常变化，获得全部权重。

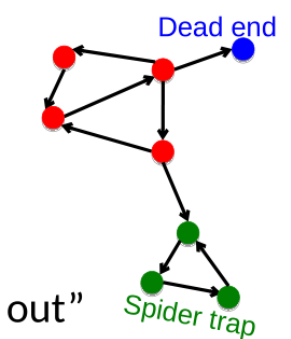


图2-8 dead ends和spider trap示意图

为了解决这个问题，我们在原有pagerank算法的基础上引入了随机游走（random walk）策略和teleport β，在一定概率下会随机跳转到另外的界面，这样能够很好的解决上述问题。我们将其用公式表示（图2-9所示），并将其化简（图2-10所示），使其易于实现，最终的公式，其

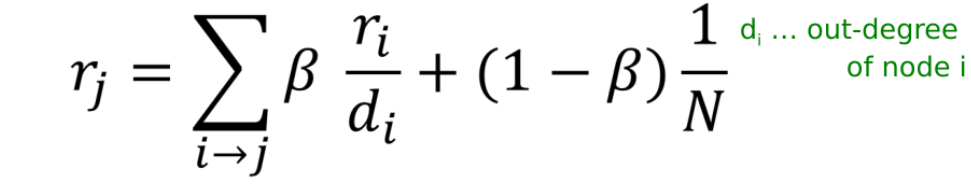


图2-9 引入teleport β的迭代公式

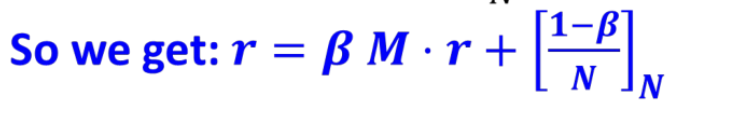


图2-10 化简后的迭代公式

相较于一般的pagerank算法，引入teleport β的迭代过程大致相同，仅仅是每次pagerank值迭代时用概率转移矩阵进行了修正。

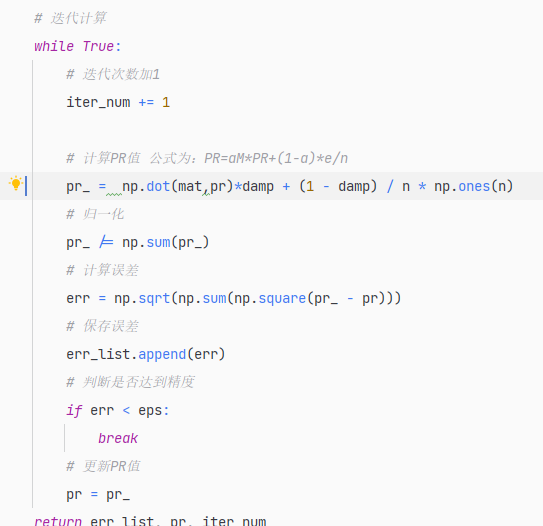


图2-11引入teleport β的pagerank算法代码展示

### 1.3.3 实验测试

**1）实验环境**

本实验使用python进行开发，版本为3.11.2。使用的库为numpy，pandas等。

实验的目录设置和基本参数如图2-12所示，默认精度（eps）为10-8，默认阻尼系数（damp）为0.85。



图2-12 实验基本参数设置

**2）PageRank算法测试**

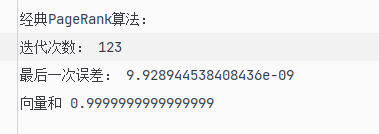


图2-13 pagerank算法运行结果截图

**3）引入teleport的PageRank算法测试**

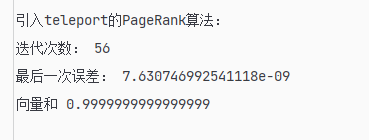


图2-14 引入teleport的pagerank算法运行结果截图

测试结果表明，pagerank算法运行正常，能够顺利收敛，同时在引入teleport β机制后，误差下降速度较快，整体收敛速度提升明显。

### 1.3.3 对比实验及结果分析

为了更加直观的展现误差err的下降过程，我绘制了err值随迭代次数变化的曲线图（图2-15）。从图中可以看出，err值在迭代次数处在[0,20]区间时下降较快，之后趋于平稳。这可能是因为记录pagerank值的pr矩阵在初始化时全部赋值为1/n，所以在最初的迭代中，pagerank值变动会较大。

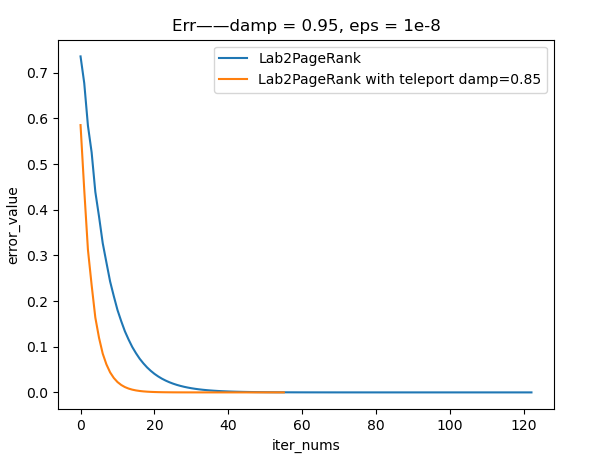


图2-15 pagerank收敛状态示意图

我注意到，teleport β机制的转移概率0.85是一个经验值，于是我探究了不同转移概率下pagerank次数的变化（图2-16）。实验中我分别将转移概率设置为0.70和0.95。从实验结果来看，随着转移概率β的提升，其更接近基础的pagerank算法，迭代次数也会增加；同时一味降低转移概率β，虽然在一定程度上降低了收敛次数，但这和本实验中pr值初始化有关，并不是一个普适的结论。事实上，将转移概率β设为0.7-0.85，pagerank都能达到不错的收敛效果。

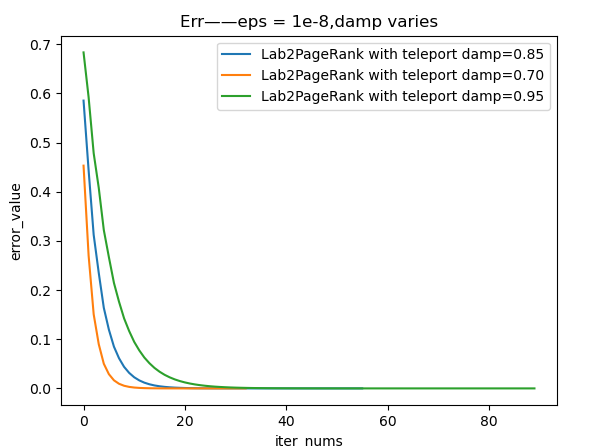


图2-16 不同的转移概率下pagerank收敛状态示意图

图2-17

## **1.4 实验总结**

### 1.4.1 实验遇到的问题及解决

**1）pr矩阵在迭代过程中未归一化**

在实验中，我获得了图2-17所展示的结果，很显然，途中的向量和（pagerank值）并不为1，经过排查，我发现在我的迭代过程中，没有对pr矩阵进行及时的归一化，导致pr值的计算越来越小，产生问题。解决方案即在每次迭代时增加归一化处理。



图2-17 错误输出截图



图2-18 解决方案

**2）矩阵读入错误**

最开始我的迭代次数高达181次，对于一个较小的数据集来说有些异常。在和同学的讨论和交流后，我发现在使用“send\_receive.csv”数据生成矩阵时，我没有考虑到有些页面没有任何的链接，直接使用person的数量作为矩阵的维度（这样算出来的维度为185\*185）。为了解决这个问题，我在代码中增加判断，得到最终的矩阵维度（182\*182）。

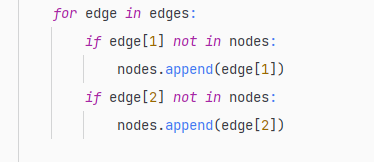


图2-19 判断某个页面是否完全没有外链接

### 1.4.2 实验总结

我一直记得老师上课时对于pagerank算法的评价。老师说这个算法正因为非常的简单易用（实现起来的确如此），同时能够保持较好的性能，从而得到了广泛的应用。这样“简单就是美”的设计理念让我陶醉。通过实验我也深入了解了pagerank算法的数学原理，迭代过程，亲手实现了一个小pagerank。

同时，我还关注到，pagerank算法并不完美，或许大部分算法也是这样，在应用中会产生更多的现实问题，这恰恰是一个算法优化改进的动力，通过引入teleport β机制，解决了dead ends，spider trap等问题，极大地提升了可用性和泛用性。我们在设计算法时，也要在应用中不断改进，不断优化，以达到最好的效果。