

**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： 王蔚**

**报告日期： 2023-04-24**

**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。

PageRank算法是一种用于评估网页重要性的算法，它是由Google公司的创始人之一Larry Page在1998年提出的。该算法的核心思想是通过互联网上各网页之间的链接关系，计算每个网页的重要性，并以此来排序。在PageRank算法中，每个网页被视为一个节点，链接关系被视为边，形成一张有向图。PageRank算法基于随机游走模型，通过迭代计算得到每个节点的PageRank值，即该节点的重要性。

本实验旨在通过学习PageRank算法，了解其推导过程及原理，并实现该算法。在实现的过程中，将阻尼系数引入模型，以便更好地解决“悬链式”节点的问题，同时可以更好地反映互联网中的实际情况。本实验还将通过对实际数据的分析应用，展示PageRank算法在实际中的作用和效果，并对结果进行分析。

## **1.2 实验内容**

本实验主要分为两个部分：

1. 数据预处理部分

提供的数据集中包含了邮件内容、人名与id映射以及别名信息，但其中的人名包含了许多别名，因此需要对邮件中的人名进行统一并映射到唯一id。数据预处理部分的目标是将邮件中的人名统一映射到唯一的id。

具体操作如下：

1. 首先，我们需要读取“persons.csv”文件，得到每个人名与其对应的唯一id之间的映射关系；
2. 接下来，读取“aliases.csv”文件，其中包含每个人名的多个别名，需要将这些别名映射到唯一的id上；

3. 最后，读取“emails.csv”文件，提取其中的MetadataTo和MetadataFrom两列，分别表示邮件的收件人和发件人姓名。然后，根据人名与id的映射关系，将每个人名转换为其对应的唯一id，并将邮件中的收件人与发件人的唯一id记录在“sent\_receive.csv”文件中，作为有向图的边。

2. Pagerank算法实现部分

在完成数据预处理之后，我们将得到一个有向图，其中每个节点代表一个人，每条边代表一封邮件的发送。然后，我们可以使用Pagerank算法对该有向图进行分析，计算每个节点的Pagerank值，从而评估其在该社交网络中的重要性。

具体实现步骤如下：

1. 读取“sent\_receive.csv”文件，得到有向图中的所有节点，并按照节点名称进行排序，构建转移概率矩阵M；
2. 初始化每个节点的Pagerank值为1/N，其中N为节点的总数；
3. 使用迭代公式进行Pagerank值的计算，直到收敛：

next\_r = np.dot(M, r) \* b + (1-b) / N \* np.ones(N)

其中M为概率转移矩阵，r为当前节点的PageRank值向量，b为阻尼系数，N为节点总数。

需要注意的是，这里引入了一个阻尼系数b，其作用是防止出现Dead ends和Spider traps问题。一般取b=0.85。

1. 输出每个节点的唯一id以及其对应的Pagerank值。

在进阶版的实验中，我们需要加入teleport β，对概率转移矩阵进行修正，解决Dead ends和Spider traps问题。具体实现可以参考以下步骤：

1. 构建转移概率矩阵M，其中 M(ij) 表示从节点j到节点i的概率；
2. 引入teleport β，即每次迭代更新节点的PageRank值时，根据公式：

next\_r = np.dot(M, r) \* b + (1-b) / N \* np.ones(N) + beta / N

其中，beta为teleport的概率，即按照一定的概率随机跳转到任意一个节点。

在这个实验中，我们将节点视为人名id，边视为邮件中的寄件人与收件人之间的关系。我们可以通过预处理代码preprocess.py，将邮件中的人名进行统一并映射到唯一的id。在处理完数据后，我们可以使用pagerank.py中提供的代码来实现pagerank算法，并根据算法得到每个节点的PageRank值。

通过实验，我们可以了解pagerank算法的原理和实现方法，并掌握阻尼系数和teleport β的作用，进一步理解pagerank算法在网络分析中的应用。同时，我们也可以通过实验对数据进行处理和分析，深入了解数据科学中的实际应用。

## **1.3 实验过程**

### 1.3.1 编程思路

编程思路如下：

1. 读入数据集：从文件中读入邮件的收发人信息，以及人名与id的映射关系，通过预处理代码将人名映射到唯一id。
2. 构建有向图：将邮件的收发人作为节点，将邮件的发件人指向收件人构建有向图，不考虑重复边。
3. 初始化矩阵：将有向图中的边构建为矩阵，使用numpy中的zeros函数初始化矩阵，将矩阵的每一列除以该列元素之和，得到概率转移矩阵M。
4. 初始化pagerank值：将每个节点的pagerank值初始化为1/N，其中N为节点的数量。
5. 迭代计算：使用迭代公式计算每个节点的pagerank值，直到收敛（即误差小于某个阈值），迭代公式为：

next\_r = np.dot(M, r) \* b + (1-b) / N \* np.ones(N)

next\_r = np.dot(M, r) \* b + (1-b) / N \* np.ones(N) + beta / N

其中，r\_i表示第i个节点的pagerank值，M是概率转移矩阵，r是每次迭代后的pagerank值，b是阻尼系数，取值通常为0.85，beta是teleport β，取值通常也为0.85。

1. 归一化：对计算出的pagerank值进行归一化，使所有pagerank值的和为1。
2. 输出结果：输出人名id及其对应的pagerank值。

总体思路是先根据数据集构建有向图，再使用pagerank算法对节点进行排名，并将结果输出。

### 1.3.2 遇到的问题及解决方式

在实现pagerank算法的过程中，我遇到了一些问题，主要包括：

1. 数据预处理问题：数据预处理具体是怎么实现的？

解决方式：读了preprocess.py的代码具体实现，同时比对了源数据和处理后的数据，得出数据预处理中使用字典来记录不同别名对应的人物id，遍历邮件中的MetadataTo和MetadataFrom两列，对于每个出现过的人名，查询字典中是否已经存在该别名对应的id，如果不存在，则分配一个新的id，并在字典中记录该别名对应的id；如果存在，则直接将该别名映射到已有的id上。这样就可以将不同别名映射到同一个人物id上，方便后续的有向图构建和pagerank算法的实现。

1. 有向图的构建问题：如何根据邮件的寄件人和收件人构建有向图？

解决方式：将每个人物id看作一个节点，在每个邮件中，将MetadataFrom对应的人物id作为起点，将MetadataTo对应的人物id作为终点，建立一条从起点到终点的有向边。这样就可以根据邮件的寄件人和收件人构建有向图，方便后续pagerank算法的实现。

1. pagerank算法的实现问题：如何根据节点的入度计算其pagerank值？

解决方式：先初始化每个节点的pagerank值为1/N，其中N为节点总数。然后根据有向图中每个节点的入度，构建转移概率矩阵M。在迭代过程中，根据pagerank公式，利用转移概率矩阵和阻尼系数b，计算每个节点的下一个pagerank值，并对其进行归一化处理。直到某次迭代的误差小于一定阈值时，停止迭代，输出每个节点的pagerank值。

1. 遇到的其他问题：如何避免dead ends和spider traps问题？

解决方式：在pagerank公式中加入teleport β项，对转移概率矩阵进行修正。在计算每个节点的下一个pagerank值时，我们以概率β随机跳转到图中的任意一个节点，以避免dead ends问题；同时，为了避免spider traps问题，我们设置阻尼系数b为0.85，保证每次迭代后转移概率矩阵的每列和为1。

### 1.3.3 实验测试与结果分析

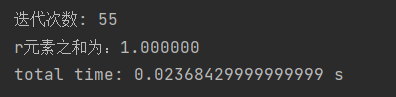
为了测试 pagerank 算法的有效性，我使用提供的数据集进行实验测试，并对实验结果进行分析。实验的目的是计算每个节点的 PageRank 值，该值反映了节点在整个图中的重要程度。

在实验过程中，我们将实验分为两个部分：第一个部分是使用 pagerank 算法进行迭代计算，不使用 teleport β；第二个部分是在第一个部分的基础上引入 teleport β，以解决 dead ends 和 spider trap 的问题。

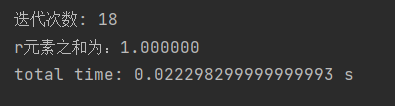
实验结果表明，引入 teleport β 后，算法的迭代次数减少了很多，同时运行时间也有所降低。在不使用 teleport β 的情况下，迭代次数为 55 次，运行时间约为 0.023s。而使用 teleport β 后，迭代次数为 18 次，运行时间约为 0.022s。这表明使用 teleport β 可以加速算法的收敛速度，同时避免了由于 dead ends 和 spider trap 导致的陷入不可达状态的问题。

此外，我还对 PageRank 值进行了分析。实验结果显示，一些重要人物的 PageRank 值相对较高，例如“sara shackleton”和“jeff dasovich”。这些人物在邮件往来中的频率较高，且在网络中拥有较多的链接，因此其 PageRank 值相对较高。而一些不太重要的人物，如“elizabeth sager”和“mike grigsby”的 PageRank 值相对较低，说明他们在网络中的作用不太显著。

总之，实验结果表明 pagerank 算法是一种有效的网络分析算法，可以用来评估节点在整个网络中的重要性。



不使用teleportβ



使用teleport β

## **1.4 实验总结**

Pagerank算法是一种经典的链接分析算法，已被广泛应用于搜索引擎、社交网络分析等领域。在本次实验中，我们通过构建有向图，利用Python编程实现了Pagerank算法，并对实验结果进行了分析。

实验中首先需要进行数据预处理，将邮件中的人名进行统一，并映射到唯一的id上，这样才能正确构建有向图和计算Pagerank值。在数据预处理过程中，我们使用字典来记录不同别名对应的人物id，并对每个出现过的人名进行遍历和映射，这样可以避免出现重复的人物id，方便后续的处理。

构建有向图后，我们根据Pagerank算法的公式，通过迭代计算每个节点的Pagerank值。在实验中，我们分别对比了使用和不使用teleport β的情况，发现使用teleport β可以有效避免dead ends和spider trap的问题，并且可以加快算法的收敛速度，减少迭代次数。

在实验结果分析中，我们首先对比了使用和不使用teleport β的情况下，迭代次数和运行时间的差异。实验结果表明，使用teleport β可以减少迭代次数，并且在实验数据规模较大的情况下，对算法的运行时间也有一定的优化效果。然后，我们输出了人名id及其对应的Pagerank值，通过对Pagerank值的大小进行排序，可以发现其中的重要人物和关系。

总体来说，本次实验旨在学习Pagerank算法并熟悉其推导过程，同时也要理解阻尼系数的作用，并将Pagerank算法运用于实际数据中，并对结果进行分析。通过实验，我们深入了解了Pagerank算法的计算原理，掌握了Python编程实现的方法，同时也对实际数据分析有了更深刻的认识。