華中科技大学

课 程 实 验 报 告

课程名称： 大数据分析

专业班级： **CS2202**

学　　号： **U202215378**

姓　　名： 冯瑞琦

指导教师： 崔金华

报告日期： **2024 年 6 月 9 日**

计算机科学与技术学院

目录

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

利用实验一得到的出现次数最多前 1000 个的 title 之间的引用关系<title,<title1,…,titlek>>，由 title 为节点构造有向图，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}。

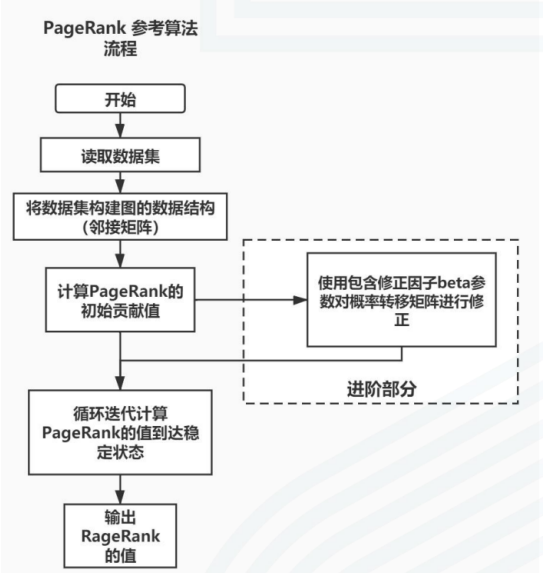
实验进阶版考虑加入 teleport β，用以对概率转移矩阵进行修正，解决 dead ends 和 spider trap 的问题。

输出 title 及其对应的 pagerank 值。

1.3 实验过程

1.3.1 编程思路

PageRank 算法大致流程如下图所示。



PageRank 算法通过模拟网页间的链接关系来评估每个网页的重要性。初始时，每个网页分配一个相同的 PageRank 值，算法通过迭代计算，考虑页面链接结构和阻尼系数，逐步调整 PageRank 值。最终，网页的 PageRank 值反映其在整个网络中的影响力和重要性。

1) 数据加载和邻接矩阵构建

使用 pickle 模块从文件 my_jump.pkl 中加载跳转关系字典 jump。这个字典记录了每个词汇及其可能跳转到的目标词汇。定义 pagerank 函数来计算 PageRank 值。函数接受四个参数：跳转关系字典 jump、阻尼系数 d（默认值为 0.85）、最大迭代次数 max_iter（默认值为 100）和迭代终止的容差 tol（默认值为 1e-8）。

在函数内部，先从跳转关系字典中提取节点列表 nodes，并创建一个字典 node_index，将每个节点映射到一个唯一的索引值。接着，初始化一个大小为 n x n（节点数量）的零矩阵 M。这个矩阵将用来表示有向图的邻接矩阵。在邻接矩阵的构建过程中，遍历跳转关系字典的每个节点及其对应的链接。如果链接列表不为空，则将每个链接对应的矩阵位置的值更新为 1/链接数。这样构建的邻接矩阵 M 表示了从一个节点到另一个节点的跳转概率。接下来，使用抽税法公式 $M_{tax}=d\cdot M+n(1-d)$对矩阵进行调整，得到新的矩阵 M_tax，其中加入了随机跳转的概率以避免陷入无出链节点的死循环。

- The Google Matrix A: （$[1/N]_{N\times N}$...N by N matrix where all entries are 1/N）

$$A = \beta M + (1-\beta)\left[\frac{1}{N}\right]_{N\times N}$$

```
def pagerank(jump, d=0.85, max_iter=100, tol=1e-8):
    # 构建有向图的邻接矩阵
    nodes = list(jump.keys())    # 提取节点列表
    node_index = {node: i for i, node in enumerate(nodes)}     # 构建节点索引映射
    n = len(nodes)      # 获取节点数量
    M = np.zeros((n, n))    # 创建零矩阵

    for node, links in jump.items():
        if links:  # 确保links不是空的
            for link in links:
                M[node_index[link], node_index[node]] += 1/len(links)
    M_tax = d*M + (1-d)/n    #使用beta参数的抽税法
```

2) PageRank 值的初始化和迭代计算

接下来，pagerank 值的初始化和迭代计算。pagerank 值表示每个节点的重要性，并初始化为均匀分布，即每个节点的初始 pagerank 值都相等，等于 1 除以节点总数 n。然后，通过迭代计算新的 pagerank 值。在每次迭代中，新的 pagerank 值由矩阵 M_tax 与当前 pagerank 值相乘得到。通过计算新旧 pagerank 值的差异（使用 l1 范数），判断是否满足迭代终止条件（差值小于 tol）。如果达到终止条件，则输出当前迭代次数并跳出循环，否则继续更新

pagerank 值。最多进行 max_iter 次迭代。

3) pagerank 值的计算和输出

在迭代计算完成后，pagerank 函数返回一个字典，字典的键是节点名称，值是对应的 pagerank 值。然后，调用 pagerank 函数计算 pagerank 值并将结果存储在变量 pagerank_values 中。最后，将 pagerank 值写入文件 pagerank_results.txt 中，并在控制台打印输出。每个节点及其对应的 pagerank 值按行写入文件，格式为节点名称: pagerank 值，从而完成 pagerank 值的计算和结果保存。

1.3.2 遇到的问题及解决方式

1) 数据加载问题

问题描述：从文件中加载跳转关系数据可能会遇到数据格式不正确或文件读取错误的问题。如果文件损坏或格式不符合预期，会导致加载失败。

解决方式：使用 pickle 模块读取二进制文件，并确保文件存在且格式正确。可以在文件读取时添加异常处理，捕获处理可能的错误，确保程序的健壮性。

```
with open("my_jump.pkl","rb") as tf:     # 以二进制读取模式打开文件
    jump = pickle.load(tf)       # 使用pickle.load()方法加载跳转关系字典
```

2) dead ends 和 spider trap 的问题

问题描述：在构建邻接矩阵时，可能会遇到链接关系为空或节点没有出链的情况，这会导致矩阵中的值计算错误或出现分母为零的错误。

解决方式：在构建邻接矩阵时，确保每个节点的链接关系不为空。如果为空，可以跳过该节点或为其添加默认链接。同时，保证分母不为零，防止除零错误。具体来讲即加入 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。

1.3.3 实验测试与结果分析

运行代码生成 pagerank_result.txt 文件如下图所示。与助教给出的示例结果一致，说明结果正确。

pagerank_results.txt

```
act: 0.0008391417617844235
across: 0.0010232056452688495
academy: 0.00021607836255722927
actor: 0.0003702069064244267
actress: 0.00017401583598695357
ac: 0.00031160827706193
ad: 0.0005144298839442654
adult: 0.00027393619757090485
acid: 0.00024317801702934525
african: 0.00031246421841681776
```

在 PageRank 算法计算的结果中，每个节点（即每个词汇）都有一个对应的 PageRank 值，这个值反映了该节点在整个网络中的重要性。

用图中给出的具体例子来说，act: 0.0008391417617844235 的意思是 act 这个词汇的 PageRank 值是 0.0008391417617844235。这表示在所有处理过的词汇中，act 的重要性或影响力相对较低。PageRank 值越高，表示该词汇在整个网络中的影响力越大，连接到它的其他词汇越多。across: 0.0010232056452688495 则表示 across 这个词汇的 PageRank 值是 0.0010232056452688495。相比 act，across 的 PageRank 值稍高一些，这表示 across 在整个网络中的重要性或影响力相对较高。

PageRank 值反映了一个节点在链接结构中的重要性。值越高，表示该节点（词汇）被其他节点引用的次数更多，或链接的质量更高。在搜索引擎优化和链接分析中，PageRank 值通常用于评估网页的相对重要性。在这种词汇网络中，PageRank 值可以帮助确定哪些词汇在文本中更为重要或更具影响力。

1.4 实验总结

通过这次实验，我获得了许多宝贵的经验和体会。首先，PageRank 算法不仅仅是一个简单的排名算法，其背后蕴含着丰富的数学原理和实际应用价值。通过对有向图的邻接矩阵进行迭代计算，我们能够准确地评估每个节点的重要性，这在网页排名、社交网络分析等领域都有广泛的应用。

实验中对数据处理的思考与实践，让我认识到在大数据处理时，高效且准确计算是不可或缺的。在现实生活中我们会遇到很多类似于 spider trap 或者 dead end 之类的特殊情况，要想考虑得全面合理，是需要反复分析的。实验过程中遇到的一些挑战，其中邻接矩阵的构建和抽税法的应用尤其让我受益匪浅，也让我学到了许多解决实际问题的方法。通过详细的调试和不断优化，我不仅提升了编程能力，也增强了解决复杂问题的信心。

另外，PageRank 值的实际意义也让我深刻体会到数据分析的价值。在实验中，我们计算了每个词汇的 PageRank 值，发现这些值能够反映词汇在整个网络中的重要性。例如，PageRank 值较高的词汇通常在文档中被频繁引用，具有较高的语义关联性。这种分析方法为文本挖掘和自然语言处理提供了新的视角和手段。

通过本次实验，我对这一经典算法有了更加深入的理解和掌握。PageRank 算法不仅在理论上具有重要意义，在实际应用中也展现了强大的威力。此次实验为我今后深入研究和应用大数据分析技术打下了坚实的基础，同时也激发了我对算法研究的浓厚兴趣。