南開大學
Nankai University

计算机学院

算法导论报告文档

基于图邻接表的 PageRank 算法

姓名：郭允轩

专业：计算机科学与技术

2025 年 6 月 7 日

目录

1 问题描述

上个世纪 90 年代，互联网迎来了属于自己的春天。在 1990 年底，伴随着任职于欧洲核子研究组织的蒂姆 · 伯纳斯-李推出世界上第一个网页浏览器和第一个网页服务器，推动了万维网的产生，直接导致了互联网应用的迅速发展，网页数量也因此呈指数级爆炸增长。海量的网页，为如何进行有效且精准的查询、如何有效评估网页的“重要性”和“权威性”，提出了一个难题。

传统的方法是基于**关键词匹配**的排序方法，根据关键词在网页中的匹配程度、位置、出现频次等对网页进行评分，它的缺点在于难以反映网页之间的结构性关系，难以有效评估出重要相关网页。

设想一下，如果现在有两个文章，其中一个文章质量不高，但是很符合匹配要求 (标题、正文中关键词出现很多次)；另外一个文章匹配程度并不如前者，但是它质量很高，被很多人引用，是这方面的权威重要文章。对于这两个文章，使用关键词匹配的方法，那么第一个文章就会被排在前面。因此存在某些网页来回地倒腾某些关键词使自己的搜索排名靠前 [1]。更何况，在海量网页涌现下，相关文章更是多如牛毛，很难有效筛选出真正质量高的网页。

所以，传统方法不能解决上面提到的问题。好在魔高一尺道高一丈，我们总有办法提出有效的算法解决计算类的难题。1997 年，**PageRank** 算法横空出世。

2 算法背景

PageRank, 即网页排名，又称网页级别、Google 左侧排名或佩奇排名，是 Google 创始人拉里 · 佩奇和谢尔盖 · 布林于 1997 年构建早期的搜索系统原型时提出的链接分析算法，自从 Google 在商业上获得空前的成功后，该算法也成为其他搜索引擎和学术界十分关注的计算模型。目前很多重要的链接分析算法都是在 PageRank 算法基础上衍生出来的。PageRank 是 Google 用于用来标识网页的等级/重要性的一种方法，是 Google 用来衡量一个网站的好坏的唯一标准。[2]

PageRank 算法提出的排序方式是这样的：分析网页之间的超链接关系 [3]，从图结构的角度出发，判断网页的“被引用”程度，从而评估其相对重要性。可以这样理解，一个网页被更多的高质量网页链接，其重要性就越高。

它的计算过程如下所示：

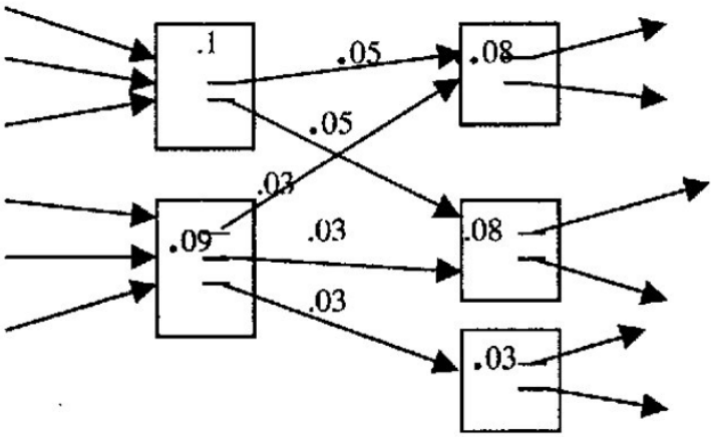

图 2.1: PageRank 计算图示

3 项目介绍

本项目基于 C++ 语言，使用构造的网页链接图 (txt 文件) 对 PageRank 算法进行过程复现和结果测试，从而验证 PageRank 算法在小规模网络中的有效性和收敛性。由于真实的搜索引擎网页数目过于庞大，本项目不选择直接在真实的搜索引擎上面复现。

在代码中，我们用一个**有向图**表示网页之间的链接关系，其中图的每个节点表示一个网页，边表示超链接; 采用图的**邻接表**作为数据结构来存储图，并基于 PageRank 算法，迭代计算出每个网页的评分值，最终实现对网页的排序。

项目代码目录非常简单明了:

- data 目录下放的是测试用的 txt 图描述文档。
- src 下的 pagerank.cpp 是封装好的 PageRank 算法实现。
- src 下的 main.cpp 是主函数，用来在不同的测试用例上测试 PageRank 算法并得出结果。

4 算法设计

本项目的 PageRank 算法实现分为以下几个部分：图的构建、PageRank 值初始化、迭代计算和收敛判断几个部分。下面详细说明各部分的实现思路。

4.1 图数据读取与构建

- 使用标准文件流读取图数据文件 (txt 文件)，每一行的格式是：源节点出链节点 1 出链节点 2 ...
- 采用 `map<string, vector<string>>` 数据结构存储图结构，键为源节点，值为该节点的所有出链节点 vector 数组
- 使用 `set<string>` 收集所有节点 (不重复)，后续会基于图中的全部节点做迭代
- 通过解析每行数据，构建源节点到它的所有出链目标节点的映射关系，同时维护所有的节点集合

4.2 PageRank 初始化

- 计算图中总节点数 N，作为 PageRank 值初始化的依据
- 为每个节点分配初始 PageRank 值 $\frac{1}{N}$，保证初始状态下所有节点的重要性均等
- 使用两个 `map<string, double>` 结构分别存储当前迭代和下一次迭代的 PageRank 值

4.3 迭代计算过程

迭代计算是 PageRank 算法的核心，主要步骤如下：

- 每次迭代开始时，先计算阻尼因子带来的基础值 $\frac{1-d}{N}$，其中 d 为预设的阻尼因子，默认 $d = 0.85$。
- 遍历图中的每个源节点，将其当前 PageRank 值均匀分配给所有出链节点：
  - 计算每个出链节点应得的部分：$\frac{PR(from)}{|tos|}$
  - 将 $d \times share$ 加到目标节点的 PageRank 值中
- 处理" 悬挂节点"（无出链节点）时，其 PageRank 值会通过 $\frac{1-d}{N}$ 项保留部分重要性

4.4 收敛判断

- 每次迭代完成后，计算新旧 PageRank 值的绝对差之和 $diff$，若小于预设的容忍阈值 tol(默认 $tol = 10^{-6}$) 时，认为算法收敛并终止迭代。存在理论证明最终 PageRank 值会趋于稳定。
- 设最大迭代次数 $max\_iter = 100$，防止无限循环；最终的 PageRank 值即各节点的重要性评分

4.5 伪代码

Algorithm 1 PageRank 算法完整实现

Input: 图文件 filename, 阻尼因子 d=0.85, 最大迭代 max_iter=100, 收敛阈值 tol=1e-6

Output: 各节点 PageRank 值 pr

```
// 1. 图数据读取与构建
graph ← {}, nodes ← {}
for line ∈ readLines(filename) do
    from, tos ← parseLine(line)
    nodes ← nodes ∪ {from} ∪ tos
    graph[from] ← tos
end for
// 2. 初始化 PageRank 值
N ← |nodes|, pr ← {}
for node ∈ nodes do
    pr[node] ← 1.0/N
end for
// 3. 迭代计算
for iter ← 1 to max_iter do
    new_pr ← {}, diff ← 0.0
    for node ∈ nodes do
        new_pr[node] ← (1 − d)/N // 阻尼项初始化
    end for
    for (from, tos) ∈ graph do
        share ← pr[from] / |tos|
        for to ∈ tos do
            new_pr[to] ← new_pr[to] + d × share
        end for
    end for
    // 4. 收敛检查
    for node ∈ nodes do
        diff ← diff + |new_pr[node] - pr[node]|
        pr[node] ← new_pr[node]
    end for
    if diff < tol then break

        return pr
```

5 算法分析

5.1 时间复杂度分析

PageRank 算法的时间复杂度主要由以下部分组成：

- **图构建阶段**：$O(E+V)$，其中 E 为边数，V 为节点数。求和是因为需要遍历所有的边和节点，构建邻接表。
- **迭代计算阶段**：
  - 单次迭代成本：$O(E+V)$

$$T_{\text{iter}} = \underbrace{V}_{\text{初始化阻尼项}} + \underbrace{E}_{\text{PR 值分配}} + \underbrace{V}_{\text{收敛检查}} \tag{1}$$

  - 总时间复杂度：$O(N(E+V))$，其中 N 为实际迭代次数，所以最坏情况是迭代达到了我们人为设定的最大阈值 max_iter 时，复杂度为 $O(max\_iter \cdot (E+V))$，本项目默认取 100 轮。

5.2 空间复杂度分析

- **图存储**：使用邻接表需 $O(E+V)$ 空间
- **PageRank 向量**：存储当前和下一轮 PR 值需 $O(2V)$ 空间
- **总体空间复杂度**：$O(E+3V) \approx O(E+V)$

5.3 收敛性分析

- **收敛条件**：当两次迭代的 PR 值变化 $diff < tol$ 时终止
- **收敛速度**：

$$diff^{(k)} \approx d^k \cdot diff^{(0)} \tag{2}$$

  其中 d 为阻尼因子，理论收敛速度为线性收敛

- **阻尼因子影响**：
  - d 值越大（接近 1），收敛越慢但更能反映网络拓扑
  - d 值越小，收敛越快但更趋近于均匀分布

5.4 优缺点分析

- **优点**：PageRank 除了能解决本文开头提到的问题外，它还是一个与查询无关的静态算法，所有网页的 PageRank 值通过离线计算获得，有效减少在线查询时的计算量，极大降低了查询响应时间。[1]
- **缺点**：容易忽略搜索的主题特征；对新页面不友好 (很少被出链链接)。

指标	复杂度
时间复杂度	$O(K(E+V))$
空间复杂度	$O(E+V)$
收敛迭代次数	$O(\log_{1/d}(\frac{1}{tol}))$

表 1: PageRank 算法复杂度总结

6 实验评估

6.1 实验设置

实验使用了分别包含 6、10、20 和 50 个节点的四个 txt 文件，每个 txt 文件中的每行的形式都是：源节点 出链节点 1 出链节点 2 ...

main 函数中，对四种数据同时进行了测试 (文件开头给出了 windows 系统下的编译和运行指令，可以一键复现该实验的所有结果)。结果的部分截图如下：

```
E : 0.035625
F : 0.025
---------------------------------
test: 2, PageRank:
A : 0.181228
E : 0.113941
G : 0.108887
D : 0.108266
F : 0.103195
B : 0.098631
C : 0.098631
J : 0.0670855
I : 0.0612771
H : 0.058858
---------------------------------
test: 3, PageRank:
A : 0.304355
B : 0.0966315
C : 0.0966315
D : 0.0966315
I : 0.0661896
```

图 6.2: 运行结果

6.2 实验结果展示

为了在有限的篇幅下获得最好的展示度，我们挑选 20 个节点 PageRank 结果进行可视化。首先，我们统计每个节点的 “被链接数”，它等同于网页的 “被引用数”，决定着网页的 PageRank 值。20 个节点安装字母顺序 (A T) 依次被链接数是：12,2，2，2，2，2，2，2，4，3，1，1，1，1，1，0，0，1，1，2。

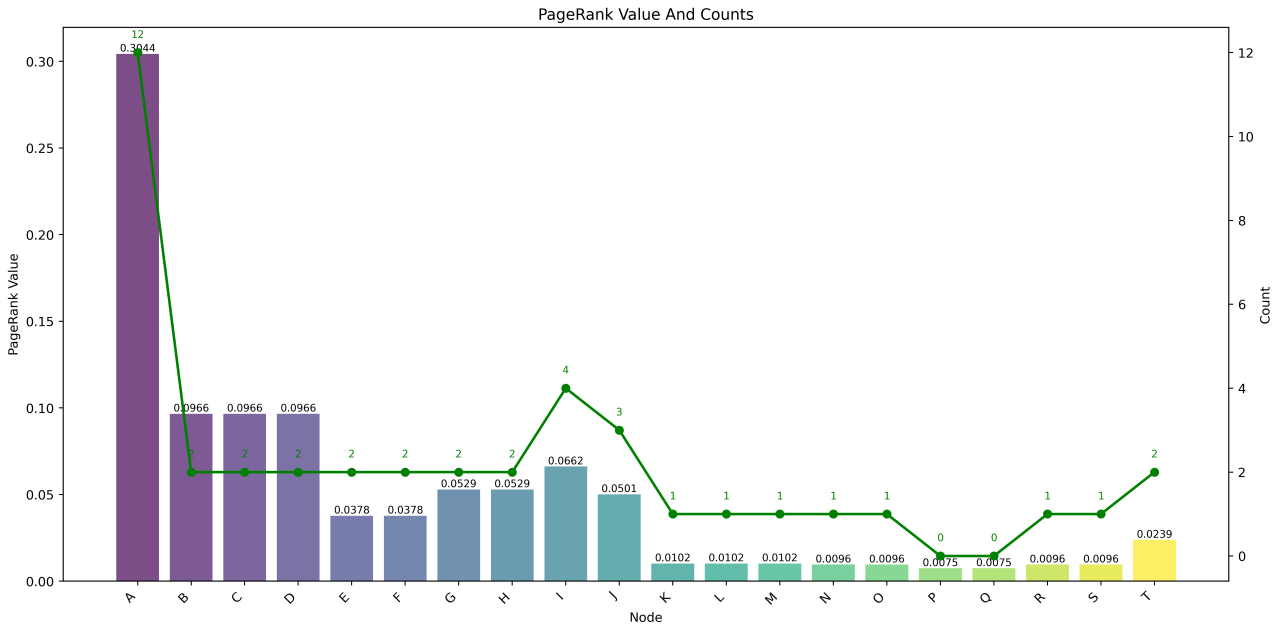

图 6.3: 结果展示

观察数据，发现被链接次数最多 (12 次) 的 A 理所应当的 pagerank 值最高。那为什么 B、C、D 仅被链接两次，pagerank 值也不低呢 (高于被链接 4 次的 I)? 打开 graph_20.txt 发现，第一行数据是 A B C D，说明 A 有链接指向了 BCD 节点,BCD 就会相应地享受到了 A 的高 rank 的部分传递，pagerank 值也就高了；其他的节点基本上 pagerank 值和被链接次数呈现正相关，并且我们的设置让被引用次数为 0 的 P、Q 也有一定的 pagerank 值 (不为 0)。ABCD 这个现象形成了小的” 推荐圈”。

7 结语

PageRank 算法是搜索引擎发展史上的一个重要的算法。本项目的思路也来源于我本学期实现的南开搜索引擎 (https://github.com/GYunnnnnX/WebSearchEngine)，被启发后，本人用 C++ 语言实现了本项目，旨在用合适的数据量、从具体算法层面探究 PageRank 算法计算原理。真实的 PageRank 算法还有很多问题，比如上面提到的“推荐圈” 问题，几个网页来回引用，导致 PageRank 难以扩散，我们采用的方法是采用随机游走，避免遍历局限在一个圈内..... 很多后来的搜索算法都基于 PageRank 算法，在后续技术发展的更多的问题上大展神通。

参考文献

[1] ali48. 你管这叫 pagerank 算法. https://blog.csdn.net/Acx77/article/details/117046280, 2021. 网页访问：2025.6.7.

[2] hguisu. Pagerank 算法. https://blog.csdn.net/hguisu/article/details/7996185, 2012. 网页访问：2025.6.7.

[3] 张俊林. 这就是搜索引擎：核心技术详解. 电子工业出版社, jan 2012. 2012 年 1 月出版.