華東師範大學
EAST CHINA NORMAL UNIVERSITY

《人工智能的数学思维》课程报告

矩阵运算与网页排名

组　员：武泽恺 10225101429
武泽恺 10225101429
武泽恺 10225101429
武泽恺 10225101429

2023 年 4 月

摘 要

矩阵运算和线性代数是计算机科学中非常重要的数学概念，Google 作为全球最大的搜索引擎，其使用的 PageRank 算法是一种基于矩阵运算的著名算法。PageRank 算法是谷歌搜索引擎中用来对网页进行排序的算法之一，该算法通过分析网页之间的链接关系来确定网页的重要性。

在 PageRank 算法中，每个网页被视为矩阵中的一个节点。通过矩阵运算，算法可以计算每个网页的 PageRank 值，从而确定网页的重要性。

本文将介绍 PageRank 算法的原理及其与矩阵运算的关系。我们将探讨如何使用矩阵运算来计算 PageRank 值，并讨论该算法在人工智能中的应用。

关键词: PageRank，网页排名，矩阵运算，搜索引擎

目 录

插 图 目 录

第一章 引言

1.1 前言

Google 作为世界上最大的搜索引擎，在市场上占有巨大的份额。如图 1.1 所示，网站 Statcounter GlobalStats 的数据显示[1]，2023 年 3 月，在全球范围的统计下，Google 占据了搜索引擎份额的 93.18%。而它的竞争对手们——bing 占据了 2.87%，Yahoo 为 1.12%，YANDEX 为 1.02%，DuckDuckGo 为 0.52%，百度为 0.42%。可以说，Google 在搜索引擎业务上几乎占据了绝对的垄断地位。

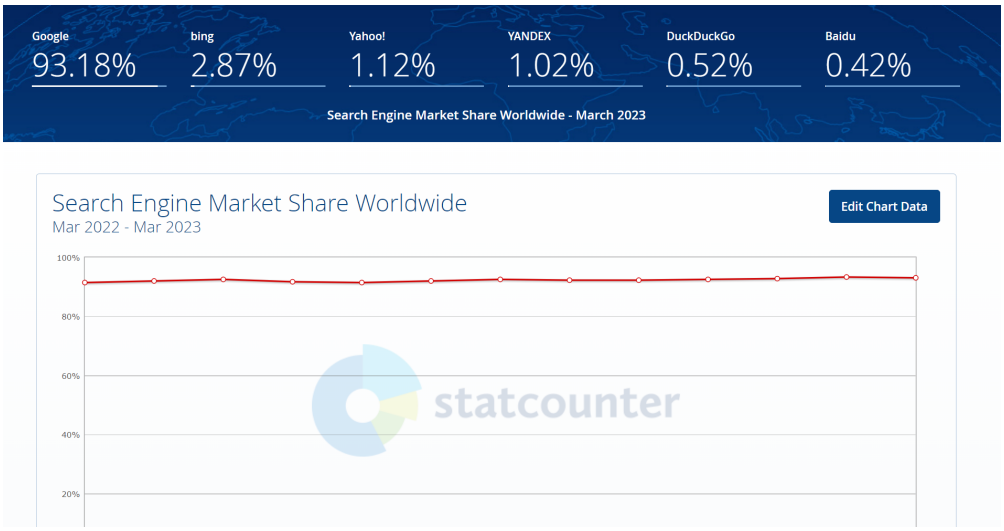

图 1.1 Statcounter GlobalStats 统计的 2023 年 3 月全球市场搜索引擎份额

而 Google 的垄断地位离不开其强大的检索功能和高质量的检索服务。

例如，我们在 Google 上搜索“大夏学堂”，得到的结果如图 1.2 所示，排名第一的是华东师范大学大夏学堂官网，排名第二的是 ECNU Forum 上关于大夏学堂相关问题的提问，排名第三的是华东师范大学课程中心关于大夏学堂停机升级的公告，接下来的则是大夏学堂优质示范课程。

而当我们在某其他搜索引擎上搜索“大夏学堂”时（如图 1.3 所示），虽然排在首位的仍是华东师范大学大夏学堂官网，排名第二的却是某小说网站上与小说《大夏学堂》有关的内容。这明显不是我们想要得到的搜索结果。

这是一个值得探究的问题。Google 是如何对搜索得到的结果进行排序，从而保障用户期望的搜索结果，即较为“重要”的网站排名靠前，而“不重要”或与搜索内容关联较小的网站排在后方？

这主要依赖排序机制。排序机制是 Google 保证搜索质量的关键。其主要依靠 PageRank 算法，该算法可以判断网页的重要性和权威度，决定其在搜索结果中的排名。Google 精准而高效的搜索结果排序，与其 PageRank 算法和对用户的深刻理解是密不可分的。这是 Google 持续领先于同业的核心因素之一，也是接下来我们要重点介绍的内容。

图 1.2 在 Google 上搜索“大夏学堂”得到的结果

图 1.3 在某搜索引擎上搜索“大夏学堂”得到的结果

1.2 涉及到的概念与方法

1.2.1 PageRank 简介

PageRank，又称网页排名、PR，是 Google 公司所使用的对其搜索引擎搜索结果中的网页进行排名的一种算法。

PageRank 本质上是一种以网页之间的超链接个数和质量作为主要因素粗略地分析网页的重要性的算法。其基本假设是：更重要的页面往往更多地被其他页面引用（或称其他页面中会更多地加入通向该页面的超链接）。其将从 A 页面到 B 页面的链接解释为“A 页面给 B 页面投票”，并根据投票来源（甚至来源的来源，即链接到 A 页面的页面）和投票对象的等级来决定被投票页面的等级。简单的说，一个高等级的页面可以提升其他低等级的页面。

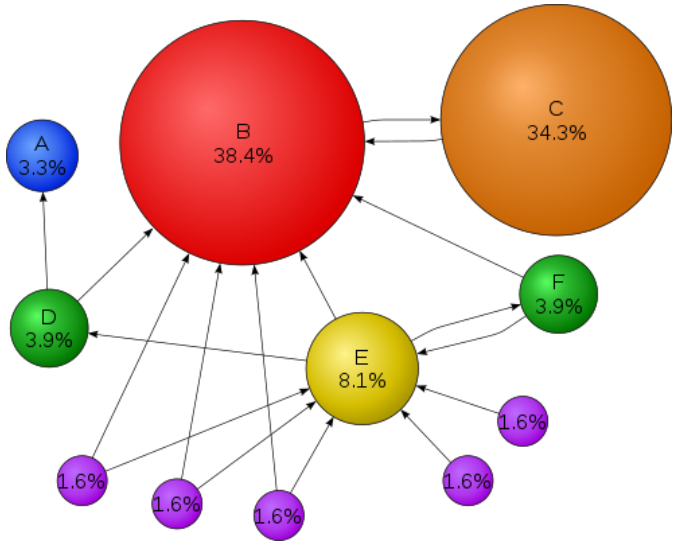

图 1.4 PageRank 算法的一个简单示例

该算法以谷歌公司创始人之一的拉里 · 佩奇（Larry Page）的名字来命名。谷歌搜索引擎用它来分析网页的相关性和重要性，在搜索引擎优化中经常被用来作为评估网页优化的成效因素之一。

目前，PageRank 算法不再是谷歌公司用来给网页进行排名的唯一算法，但它是最早的，也是最著名的算法。

1.2.2 马尔可夫链简介

马尔可夫链（Markov chain），又称离散时间马尔可夫链，因俄国数学家安德烈 · 马尔可夫得名，为状态空间中经过从一个状态到另一个状态的转换的随机过程。该过程要求具备“无记忆”的性质：下一状态的概率分布只能由当前状态决定，在时间序列中它前面的事件均与之无关。这种特定类型的“无记忆性”称作马尔可夫性质。马尔科夫链作为实际过程的统计模型具有许多应用。

在马尔可夫链的每一步，系统根据概率分布，可以从一个状态变到另一个状态，也可以保持当前状态。状态的改变叫做转移，与不同的状态改变相关的概率叫做转移概率。

当马尔可夫链运行足够长时间后，无论初始状态是什么，其状态的概率分布会达到平稳分布，并且保持不变。形式上，如果马尔可夫链有 n 个状态，转移矩阵为 P，平稳分布向量为 $\boldsymbol{\pi}$，则需要满足 $\boldsymbol{\pi} = \boldsymbol{\pi}P$。

这表示平稳分布向量 $\boldsymbol{\pi}$ 是转移矩阵 P 的特征向量，对应特征值为 1。具体来说，平稳分布具有以下性质:

1. $\boldsymbol{\pi}_i \geq 0$，且 $\sum \boldsymbol{\pi}_i = 1$。即平稳分布的每个元素都非负，且和为 1，代表概率。
2. $\boldsymbol{\pi} = \boldsymbol{\pi}P$。即平稳分布向量是转移矩阵的特征向量，对应特征值为 1。这说明如果当前状态按平稳分布，那么过一步转移后，状态的分布仍然是平稳分布。
3. 不依赖初始状态。无论初始状态是什么，经过足够长时间转移，马尔可夫链的状态分布会达到平稳分布。
4. 唯一性。马尔可夫链的平稳分布是唯一的。

平稳分布表示马尔可夫链在长期运行后的稳定态，不会再被初始状态影响。它衡量了每个状态在整个链中出现的频率，给出了马尔可夫链稳定运行时每个状态上花费的时间比例。

假设我们在上网浏览页面时，选择下一个界面的过程，与过去浏览过哪些界面没有关系，而仅依赖于当前的页面。这是一个简单的有限状态、离散时间的随机过程，可以使用马尔可夫链来描述。

1.3 本报告的主要结构和内容

本报告分为六章，将主要介绍 PageRank 网页排名算法的背景、详细内容、应用、缺陷和优化，并简单介绍其在人工智能领域的应用。

同时，本报告将重点分析线性代数（尤其是矩阵运算）以及其他数学思维在 PageRank 算法中发挥的作用，并给出具体的 PageRank 计算实例。

第二章 PageRank 算法的详细介绍

2.1 背景

PageRank 算法与矩阵的特征值有关。在 PageRank 算法被提出之前，特征值问题就在许多评分问题中被重用。许多人独立地提出使用特征值问题，包括 Edmund Landau 在 1895 年关于决定国际象棋锦标赛获胜者的建议、Gabriel Pinski 和 Francis Narin 在 1976 年关于科学计量学排名科学期刊的工作、Thomas Saaty 在 1977 年提出的层次分析过程概念中加权替代选择、以及 Bradley Love 和 Steven Sloman 在 1995 年作为概念的认知模型，即中心性算法。此外，Robin Li 于 1996 年开发的 RankDex 搜索引擎使用了类似 PageRank 的策略来评分和排名网站。

Google 的创始人 Larry Page 和 Sergey Brin 于 1996 年在斯坦福大学开发了 PageRank 算法作为一项新型搜索引擎的研究项目的一部分。他们以 PageRank 为基础创建了 Google 搜索引擎，而 PageRank 算法依然是 Google 搜索工具的基础之一。

2.2 基本原理

PageRank 算法的基本原理与矩阵运算有关。

首先，假设互联网上共有 n 个页面，可以按照如下方式定义一个 $n \times n$ 的超链接矩阵 P。假设页面 i 拥有 $k > 0$ 个指向其他页面的超链接，则如果 i 和 j 之间有这样的一条超链接，$P_{ij} = 1/k$，否则 $P_{ij} = 0$，即

$$P_{ij} = \begin{cases} 1/k, & \text{存在由 } i \text{ 指向 } j \text{ 的超链接} \\ 0, & \text{其他情况} \end{cases} \tag{2.1}$$

为了使此超链接图联通，可以假设浏览者以一定的概率访问均匀分布的任意网页。因此，可以将其定义为马尔可夫链的平稳分布，其状态空间为所有网页的集合，转移矩阵为

$$\widetilde{P} = cP + \frac{(1-c)}{n}E \tag{2.2}$$

其中，E 是一个元素全为 1 的矩阵，n 是网页页面的数量，$c \in (0,1)$ 是模型参数，我们将其称为“阻尼因子”，它表示用户不会终止当前界面的浏览且随机跳转到任意界面的概率，通常取 0.85[2]。

由于矩阵 $\widetilde{P}$ 是随机矩阵，非周期性且不可约，根据马尔科夫链的理论[3]，存在唯一的平稳分布向量 $\boldsymbol{\pi}$，使得它是转移矩阵 $\widetilde{P}$ 的特征向量，对应特征值是 1[4]。即

$$\boldsymbol{\pi}\widetilde{P} = \boldsymbol{\pi}, \quad \boldsymbol{\pi}\boldsymbol{e} = 1 \tag{2.3}$$

其中，$\boldsymbol{e}$ 为元素全为 1 的列向量。

我们将满足式 (2.3) 的向量 $\boldsymbol{\pi}$ 称为 PageRank 向量，如果一个浏览者以概率 c 跟随网页上的一个超链接，而以概率 $(1-c)$ 跳转到一个随机界面，那么 $\boldsymbol{\pi}_i$ 可以被解释为该浏览者在页面 i 的平稳概率。

最终，得到的 PageRank 向量的各分量即为各网页的 PageRank 值。

实际上，每次计算得到的 PageRank 值也可以用另一个公式

$$PR(A) = \frac{1-c}{n} + c\sum_i \frac{PR(T_i)}{L(T_i)} \tag{2.4}$$

来描述。其中，A 为所求节点，$PR(T_i)$ 为其他节点（存在指向 A 的超链接的节点）的 PageRank 值，$L(T_i)$ 为其他节点的指向其他页面的超链接数（即式 (2.1) 中的 k 值）。不难发现，公式 (2.2) 和 (2.3) 为公式 (2.4) 的矩阵表示。

2.3 计算方法

PageRank 可以通过迭代法或代数法计算。迭代法可以视为幂迭代法或幂方法。两种方法执行的基本数学运算是相同的。代数法是通过构建并解决一个描述网页关系的线性方程组来计算 PageRank 值的。这需要构造一个非常大的矩阵, 并且解方程组也很困难, 计算复杂度高。迭代法顾名思义就是通过重复迭代逼近最终解。它的基本思想是: 每个网页的 PageRank 值都等于其他所有指向它的网页的 PageRank 值之和除以对应网页的出链总数。通过多次迭代计算,PageRank 值会逐步收敛至稳定。

因此，在本部分，我们仅介绍迭代法。

2.3.1 迭代原理

我们使用 $PR(x,i)$ 来表示页面 x 在第 i 次迭代时得到的 PageRank 值。在 $i=0$ 时，我们需要对各网页的 PageRank 值进行初始化，通常取

$$PR(P_i, 0) = \frac{1}{n} \tag{2.5}$$

接下来，根据公式 (2.4)，我们对 PageRank 值进行迭代计算

$$PR(P_i, i+1) = \frac{1-c}{n} + c\sum_i \frac{PR(P_j, i)}{L(P_i)} \tag{2.6}$$

或根据公式 (2.2) 和 (2.3) 使用矩阵表示

$$\boldsymbol{\pi}_{i+1} = \boldsymbol{\pi}_i \widetilde{P} = cP\boldsymbol{\pi}_i + \frac{1-c}{n}\boldsymbol{e} \tag{2.7}$$

根据公式 (2.6) 或公式 (2.7) 我们可以通过重复迭代计算至其收敛，满足

$$|PR(P_i, i+1) - PR(P_i, i)| < \epsilon \tag{2.8}$$

其中，ϵ 是一个很小的值。此时的 PageRank 值即可视为网页的最终 PageRank 值。

2.3.2 幂迭代法

设超链接矩阵为 P，向量 $\boldsymbol{\pi}$ 为平稳分布向量（即其满足公式 (2.3)），则有

$$\boldsymbol{\pi} = (cP + \frac{1-c}{n}E)\boldsymbol{\pi} = \widetilde{P}\boldsymbol{\pi} \tag{2.9}$$

故 $\boldsymbol{\pi}$ 为矩阵 $\widetilde{P}$ 的特征向量。

因此，便存在一种快速简便的计算方法——幂迭代法：从任意向量 $\boldsymbol{x}(0)$ 开始，不断地与矩阵 $\widetilde{P}$ 进行运算

$$\boldsymbol{x}(t+1) = \widetilde{P}\boldsymbol{x}(t) \tag{2.10}$$

直至

$$|\boldsymbol{x}(t+1) - \boldsymbol{x}(t)| < \epsilon \tag{2.11}$$

此时，向量 $\boldsymbol{x}(t)$ 即为 PageRank 向量 $\boldsymbol{\pi}$。

2.3.3 代码实现

可以使用 Python 实现上述幂迭代法。

```
import numpy as np
def pagerank(P: np.array, num: int = 100, c: float = 0.85):
    P = np.transpose(P)
    N = P.shape[1]
    v = np.ones(N) / N
    P_hat = c * P + (1 - c) / N
    for _ in range(num):
        v = P_hat @ v
    return v
```

其中，参数 P 为超链接矩阵 P，参数 num 为迭代次数，参数 d 为阻尼系数 d。

第三章　简单的 PageRank 实例

3.1　实例 1：影响 PageRank 的因素

考虑一个仅由四个网页组成的简单网络，结构如下图所示。

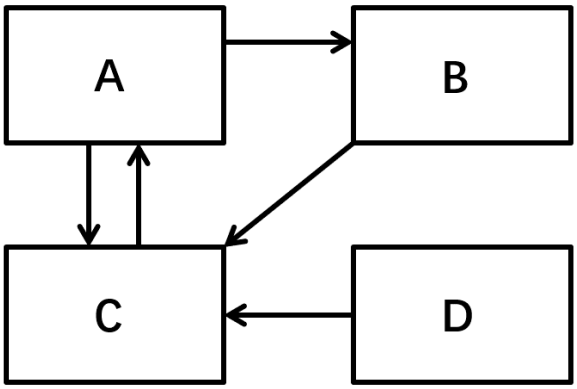

图 3.1 实例 1 中的简单网络

使用公式 (2.1) 求出其超链接矩阵

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

取阻尼系数 $c = 0.85$，计算得到其转移矩阵

$$\widetilde{P} = \begin{bmatrix} 0.0375 & 0.4625 & 0.4625 & 0.0375 \\ 0.0375 & 0.0375 & 0.8875 & 0.0375 \\ 0.8875 & 0.0375 & 0.0375 & 0.0375 \\ 0.0375 & 0.0375 & 0.8875 & 0.0375 \end{bmatrix}$$

其中，$\widetilde{P_{ij}}$ 表示从页面 i 跳转到页面 j 的概率。

设 t 为迭代次数，当 $t = 0$ 时，各网页的 PageRank 有初始值 0.25。

$$\boldsymbol{x}(0) = \begin{bmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{bmatrix}$$

经历第一次迭代后

$$\boldsymbol{x}(1) = \begin{bmatrix} 0.25 \\ 0.14375 \\ 0.56875 \\ 0.0375 \end{bmatrix}$$

可以作图表示从 $t=0$ 到 $t=1$ 时 PageRank 值的转移情况。

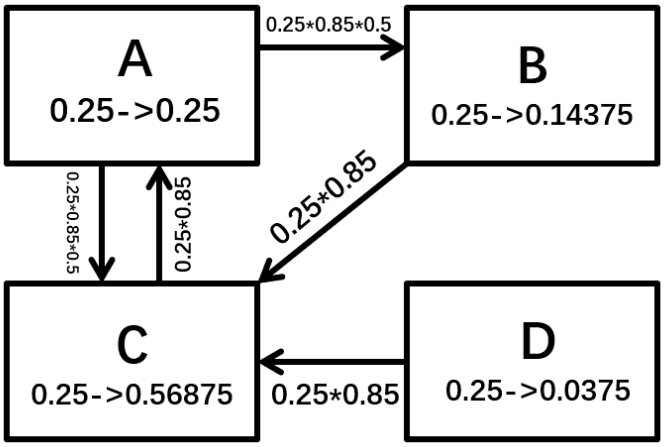

图 3.2 实例 1 中从 $t=0$ 到 $t=1$ 时 PageRank 值的转移情况

可以看出，在每次迭代的过程中，相当于将当前页面的 PageRank 值乘以阻尼系数 d，平均地通过超链接转移给目标页面[5]。

继续进行迭代，待趋于稳定时得到的 PageRank 值为

$$\boldsymbol{\pi} = \begin{bmatrix} 0.37252685 \\ 0.19582391 \\ 0.39414924 \\ 0.0375 \end{bmatrix}$$

同时也可以作出迭代过程中 PageRank 值变化的折线图。

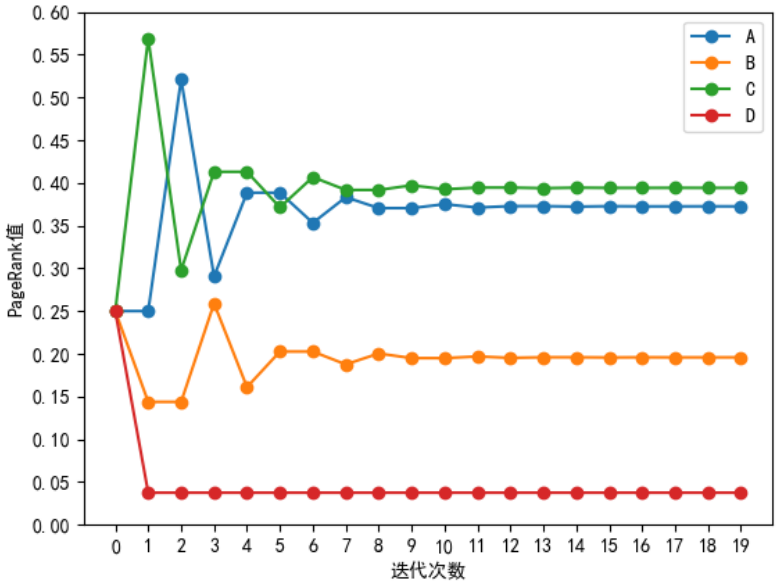

图 3.3 实例 1 迭代过程中 PageRank 值变化

因此，网页排名的最终结果为 C, A, B, D。可以发现，网页的 PageR ank 值与被链接的次数和链接它的界面的 PageRank 值有关。在这个示例中，页面 C 被链接的次数最多，其 PageRank 值最高，页面 A, B 被链接的次数相等，但由于链接到 A 的页面 C 的 PageRank 值更高，即“更优质”，A 最终的 PageRank 值高于 B。而没有页面链接到页面 D，因此其 PageRank 值最低。

3.2 实例 2：阻尼因子 c 的影响

接下来考虑阻尼因子 c 对 PageRank 值的影响。

首先，我们考虑与实例 1 相同的简单网络。

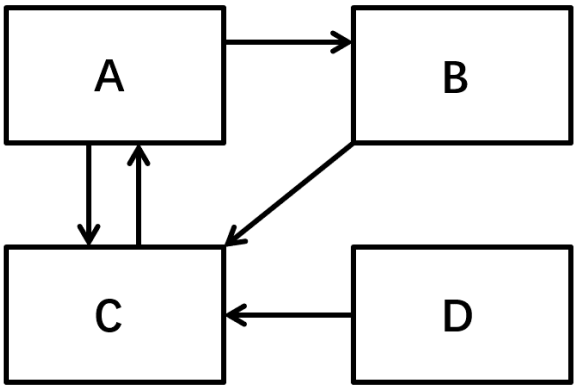

图 3.4 实例 2 中的简单网络 (1)

通过 Python 程序，我们可以作出各网页的 PageRank 值随阻尼因子 c 的取值而变化的折线图。

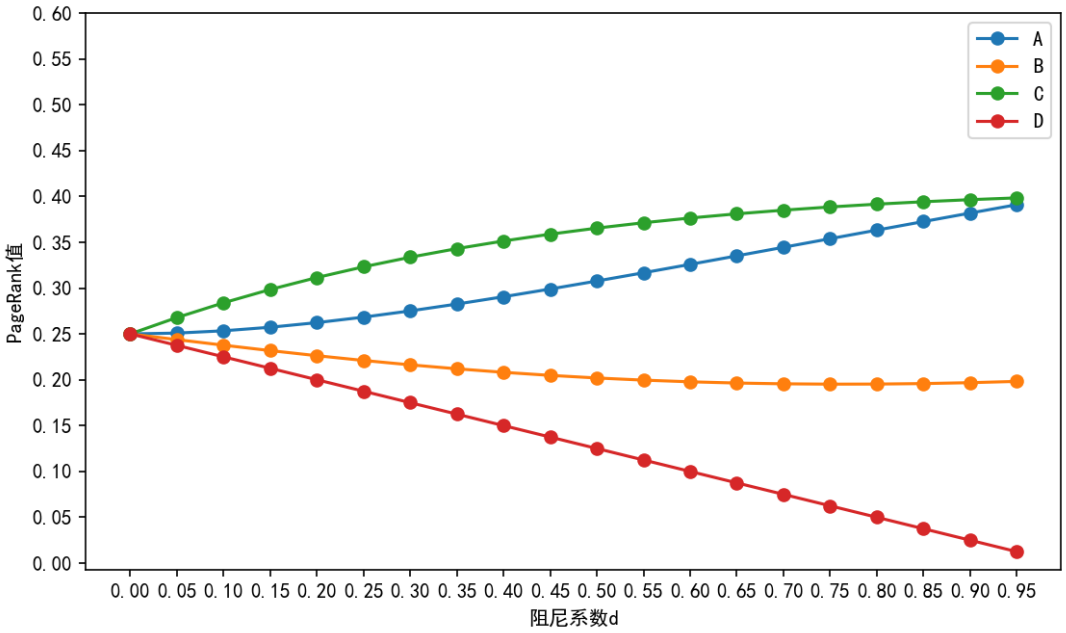

图 3.5 实例 2 中简单网络 (1) 中各网页的 PageRank 值随阻尼因子 c 的变化

可以发现，此时阻尼因子 c 越大，不同网页的 PageRank 值在大体上差异更大，网页的“优劣”更加容易区分。

接下来，我们考虑一种更简单的网络，如下图所示。

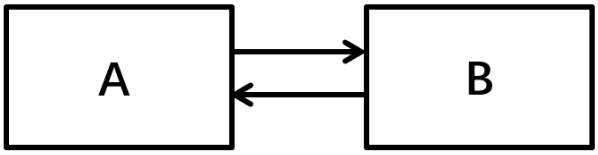

图 3.6 实例 2 中的简单网络 (2)

作出其 PageRank 值随阻尼因子 c 变化的折线图。

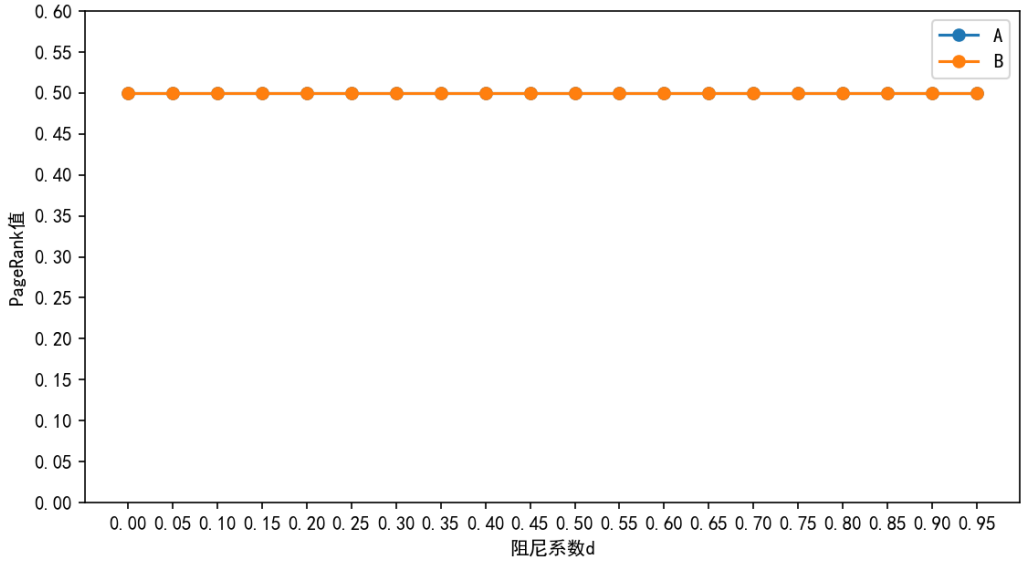

图 3.7 实例 2 中简单网络 (2) 中各网页的 PageRank 值随阻尼因子 c 的变化

不难发现，在这种情况下，其各页面的 PageRank 值与阻尼因子 c 的取值无关。

接下来，我们考虑另一种样式的简单网络，如下图所示。

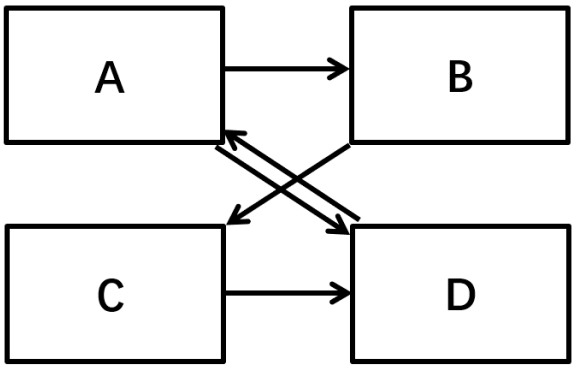

图 3.8 实例 2 中的简单网络 (3)

作出其 PageRank 值随阻尼因子 c 变化的折线图。

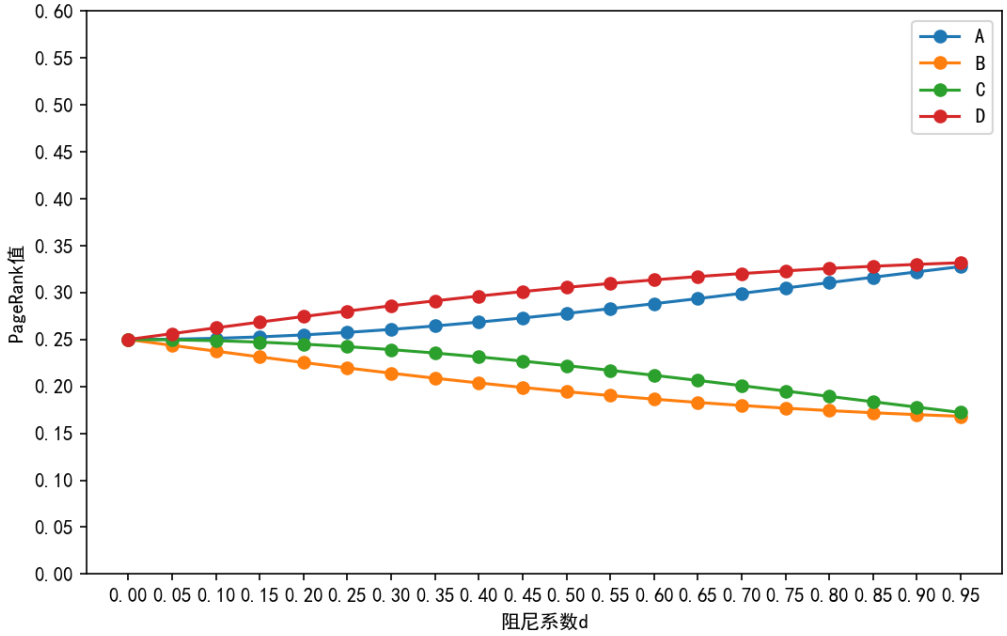

图 3.9 实例 2 中简单网络 (3) 中各网页的 PageRank 值随阻尼因子 c 的变化

可以发现，在这种情况下，阻尼因子 c 取较为靠近中间的值最为合适，过小或过大均会导致网页之间的区分度下降。

根据上述结论，阻尼因子的变动会对不同类型的网络下的不同网页差生影响，具体影响的大小和方式取决于网络的类型。由于马尔可夫链的基本假设与网页搜索的特性相同，经推到得知阻尼因子越接近 1 效果越好[6]，但在与简单网络 (3) 类型相似的网络中，阻尼因子太靠近 1 时，无法区分网页优劣，因此 Google 取阻尼因子为 0.85 必有其独特考量[2,6]。

第四章　PageRank 算法的缺陷与优化

4.1　缺陷

虽然 PageRank 算法是一种经典的网页排名算法，但它也有一些缺陷。

其中最明显的缺陷是被称为“均匀访问假设”，即假设用户访问所有网页的概率相等。这并不总是真实情况，因为用户可能更喜欢访问某些网页，例如知名网站和社交媒体网站。

易受到垃圾链接的影响也是一个严重的问题。垃圾链接是指那些存在于低质量网页或通过黑帽 SEO 手段生成的链接，它们的存在会影响到整个网页排名系统。此外，垃圾链接的使用也可能会导致 PageRank 算法的误导。例如，一个不相关的网站可以通过将大量的垃圾链接指向某个网站，来提高该网站的排名。例如，Google 搜索曾在挪威发生过一次事故，当时，大量搜索结果被某名为垃圾网站占据（“美人鱼事件”）。攻击者可以通过构建大量的虚假网页，并将这些网页与目标网站进行链接，从而提高目标网站的排名。为了应对这种风险，Google 等搜索引擎采用了各种反作弊措施，例如对网页质量进行评估、监控垃圾链接等。

另一个 PageRank 算法的缺陷是难以处理动态网页。如果一个网页的内容经常变化，PageRank 算法可能无法及时反映这些变化。例如，一些新闻网站每天都会发布大量新闻，如果 PageRank 算法只在网站更新时计算一次，那么这些新闻的重要性将无法及时反映到排名中。为了解决这个问题，一些搜索引擎采用了实时计算 PageRank 的方法。

PageRank 算法还存在数据稀疏性和主题缺乏的问题。在某些情况下，由于缺乏与搜索主题相关的网页链接，PageRank 算法可能会忽略一些重要的网页。此外，由于 PageRank 算法只考虑链接结构，而忽略网页的内容和主题，因此它可能无法很好地适应某些特定领域的搜索，例如医疗、法律等领域。

4.2　优化

4.2.1　矩阵分解的应用

在互联网上，网页的数量极为庞大，因此如何高效地计算网页的 PageRank 值成为了一个关键问题。为了跟上互联网结构的迅速变化，Google 每月更新一次 PageRank。其中一种方法便是探索超链接矩阵 P 的特定性质[4]。在本部分中，我们将利用矩阵 P 的可约性来进行优化。

考虑如下形式的超链接矩阵 P

$$P = \begin{bmatrix} P_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & P_N \end{bmatrix} \tag{4.1}$$

其中，对角线上的块 $P_I(I = 1, \ldots, N)$ 表示第 I 组内的链接。我们用 n_I 来表示第 I 组内部的网页个数。每个块 I 不与外界交流，但其本身内部可能存在着非常

复杂的关系。

接下来，我们考虑超链接矩阵 (4.1) 所对应的转移矩阵 $\widetilde{P} = cP + (1-c)(1/n)E$，令向量 $\boldsymbol{\pi}$ 为转移矩阵 $\widetilde{P}$ 的 PageRank 向量，其满足 $\boldsymbol{\pi}\widetilde{P} = \boldsymbol{\pi}, \boldsymbol{\pi}\boldsymbol{e} = 1$。此外，我们定义块 I 的摄动矩阵

$$\widetilde{P_I} = cP_I + \frac{1-c}{n_I}E \tag{4.2}$$

并令向量 $\boldsymbol{\pi}_I$ 为 $\widetilde{P_I}$ 的 PageRank 向量，使得

$$\boldsymbol{\pi}_I\widetilde{P_I} = \boldsymbol{\pi}_I, \quad \boldsymbol{\pi}_I\boldsymbol{e} = 1 \tag{4.3}$$

可以证明得到以下公式

$$\boldsymbol{\pi} = \frac{1}{n}\left(n_1\boldsymbol{\pi_1}, n_2\boldsymbol{\pi_2}, \ldots, n_N\boldsymbol{\pi_N}\right) \tag{4.4}$$

证明

我们可以证明 (4.4) 确实为 $\widetilde{P}$ 的平稳分布。定义

$$\overline{E} = \begin{bmatrix} \frac{1}{n_1}E & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{n_N}E \end{bmatrix} \tag{4.5}$$

对于式 (4.4) 中的 $\boldsymbol{\pi}$，有

$$\begin{aligned} \boldsymbol{\pi}\widetilde{P} &= \boldsymbol{\pi}\left[cP + (1-c)\overline{E} - (1-c)\overline{E} + (1-c)(1/n)E\right] \\ &= \boldsymbol{\pi}\left[cP + (1-c)\overline{E}\right] + \boldsymbol{\pi}\left[(1-c)(1/n)E - (1-c)\overline{E}\right] \\ &= \left(\frac{n_1}{n}\boldsymbol{\pi_1}\widetilde{P_1}, \ldots, \frac{n_N}{n}\boldsymbol{\pi_N}\widetilde{P_N}\right) + (1-c)(1/n)\boldsymbol{e}^T - (1-c)(1/n)\boldsymbol{e}^T \\ &= \frac{1}{n}\left(n_1\boldsymbol{\pi_1}, n_2\boldsymbol{\pi_2}, \ldots, n_N\boldsymbol{\pi_N}\right) \\ &= \boldsymbol{\pi} \end{aligned} \tag{4.6}$$

由于 $\widetilde{P}$ 的 PageRank 向量是唯一的，因此 $\boldsymbol{\pi}$ 即为 $\widetilde{P}$ 的 PageRank 向量。

该定理的证明是较为容易的。这种分解特性也可以用以下公式来表示

$$\boldsymbol{\pi} = \frac{1-c}{n}\boldsymbol{e}^T\left[1 - cP\right]^{-1} \tag{4.7}$$

接下来让我们讨论该公式对 PageRank 结果计算的意义。如果我们预先不知道超链接矩阵的块结构，则可以使用任何的图遍历算法来得到（如深度优先搜索算法和广度优先搜索算法）。图遍历算法的时间复杂度为 $O(n+m)$，其中 n 为网页的数量，m 为链接的数量。但是注意到超链接矩阵是极其稀疏的，时间复杂度在页面数量上接近线性。由于我们对超链接矩阵进行了分解，可以独立地处理所分成的每一个部分并将 PageRank 近似向量的不同部分存储在不同的数据库中，来达到节省时间和空间的目的。特别的，我们可以考虑使用尽可能多的并行处理器来进行运算，从而得到更好的计算效率。

4.2.2 主题敏感的 PageRank 算法

主题敏感的 PageRank 算法（Topic-sensitive PageRank）是 PageRank 算法的一种扩展，旨在提高搜索结果的准确性。与传统的 PageRank 算法不同，它不再只考虑页面的全局重要性，而是基于给定主题的重要性对页面进行排序。这种算法对于需要特定主题的搜索结果更为有效。

Topic-sensitive PageRank 的主要思想是根据预定义的主题构建多个 PageRank 向量，每个向量都有自己的主题，并使用它们来计算页面的权重。在常规关键字搜索查询中，可以使用查询关键字的主题计算页面的主题敏感 PageRank 分数。而在上下文搜索中（例如，当在网页中选中一些文字进行搜索查询时），可以使用查询出现的上下文的主题来计算页面的主题敏感 PageRank 分数。通过使用这些（预先计算好的）主题偏置 PageRank 向量的线性组合，可以在查询时生成特定上下文下的页面重要性分数，从而比使用单个通用 PageRank 向量生成更准确的排名结果。[7]

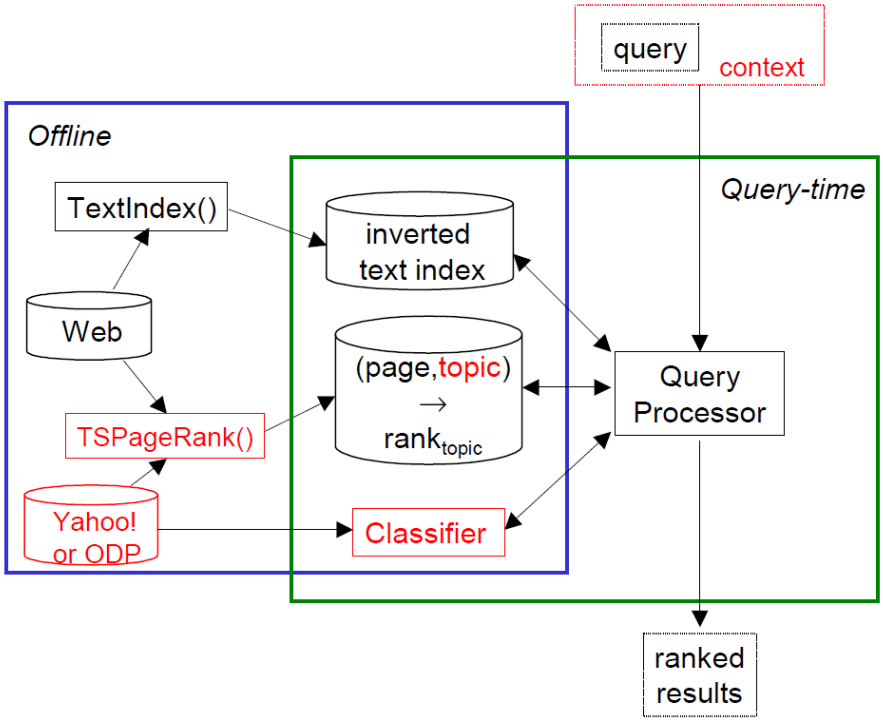

图 4.1 使用主题敏感的 PageRank 的系统示意图[7]

此方法的概述如下[7]。在离线处理 Web 爬虫期间，我们使用 Open Directory Project (ODP，一份包含超过 250 万个 URL 的 Web 目录) 中的顶级类别的 URL 生成 16 个主题敏感的 PageRank 向量。在查询时，我们计算查询（或用户上下文）与每个主题的相似度。然后，我们不使用单个全局排名向量，而是使用主题敏感向量的线性组合，加权使用查询（和任何可用上下文）与主题之间的相似度。通过使用一组排名向量，我们能够更准确地确定与特定查询或查询上下文相关的最重要的页面。由于基于链接的计算在预处理阶段进行离线处理，因此查询时间成本不比普通 PageRank 算法高。

4.2.3 稀疏矩阵优化

PageRank 算法中，最消耗计算资源的部分就是计算网页之间的链接权重，这涉及到对每个网页的出链进行求和。由于大多数网页的出链数量非常庞大，因此这个过程会产生一个非常稀疏的矩阵。为了有效地处理这个问题，可以使用稀疏矩阵优化技术。

在 PageRank 算法中，每个网页可以表示为一个节点，每个链接可以表示为从一个节点到另一个节点的边。我们可以将这些节点和边表示为一个稀疏矩阵，其中矩阵中的每个元素表示一个链接的权重。在传统的矩阵计算中，我们需要处理整个矩阵，即使其中大部分元素是 0。然而，在 PageRank 算法中，矩阵非常稀疏，因此可以使用稀疏矩阵优化技术，只处理非零元素，从而减少计算量。

稀疏矩阵优化的一种方法是使用压缩稀疏行（Compressed Sparse Row，CSR）格式。在 CSR 格式中，矩阵中的每个非零元素都保存了其值、列索引和行偏移量。行偏移量指示了每一行中第一个非零元素的位置，这使得矩阵的每一行都可以通过一个单独的数组来表示。

这种方法可以大大降低计算复杂度，因为只需要处理非零元素。在 PageRank 算法中，使用 CSR 格式可以减少计算时间和内存消耗，从而使算法可以处理更大的网页集合[8]。

第五章 PageRank 算法在人工智能中的应用

PageRank 算法最初是为了互联网搜索引擎的排序问题而提出的，但是随着人工智能的不断发展，PageRank 算法也被应用到了更多的领域。在本章节，我们将简要介绍 PageRank 算法在人工智能中可能的（或已经存在的）应用。

5.1 自然语言处理

PageRank 算法在自然语言处理中的应用比较广泛，其中一种应用是关键词提取。关键词抽取是一种自然语言处理任务，旨在从文本中提取最具代表性的关键词或短语。PageRank 算法可以用于对文本中的关键词进行排序，以确定最具有代表性的关键词。该方法使用单词之间的相似性和共现关系构建词语网络，并利用 PageRank 算法确定在该网络中最重要的单词。例如，可以基于 PageRank 算法，并结合人类语言习惯特性定义位置权重系数，实现新闻关键词提取算法[9]。

PageRank 算法还可用于文本分类任务，其中目标是将文本分为不同的类别。在这种情况下，每个类别可以视为单词网络中的一个节点。利用 PageRank 算法可以计算每个节点的权重，以便更好地理解哪些节点在文本分类中是最重要的。例如，借鉴于 PageRank 模型和跨领域倾向性分析算法，通过引入 PageRank 模型来研究词汇的原始情感倾向性识别技术，来判断情感词表现出的情感极性[10]。

5.2 聊天机器人

近期，Chat GPT 的大热让我们关注于聊天机器人和大模型。可以考虑将 PageRank（实际上即为搜索引擎）与聊天机器人相结合，以提高其智能程度。当结合 PageRank 算法和聊天机器人时，可以将聊天机器人看作是一个搜索引擎的前端。与传统的搜索引擎不同，聊天机器人可以提供更加个性化和交互式的搜索结果，从而更好地满足用户的需求。

例如，当用户输入一个问题或者话题时，聊天机器人可以使用 PageRank 算法对相关网页进行排名，并展示给用户最有用的信息。这样，聊天机器人就可以像一个个性化的搜索引擎一样，为用户提供最相关和有用的答案。而且，通过分析用户与网页的交互行为，聊天机器人可以不断学习和优化 PageRank 算法的权重和参数，提高其预测和推荐的准确性。另外，聊天机器人还可以使用 PageRank 算法来帮助用户发现与他们感兴趣的主题相关的内容。例如，当用户询问聊天机器人有关特定主题的信息时，聊天机器人可以使用 PageRank 算法来推荐与该主题相关的最佳文章或网页。

然而，目前我们无法确定结合了搜索引擎的聊天机器人 Bing AI 是否使用 PageRank 算法。不过，我们仍然可以对这个方向进行展望，结合 PageRank 和聊天机器人仍然是一个有潜力的领域。

第六章 启发和体会

[未完成]

在调研和分析 PageRank 算法的过程中，我体会到了数学思维的重要性。PageRank 算法使用了矩阵和向量运算，并利用了线性代数中的特征向量和特征值等概念。这让我深刻认识到了数学在人工智能领域的重要性，尤其是在大数据处理方面。

在了解 PageRank 算法的应用领域时，我发现其广泛应用于搜索引擎中，这使我对搜索引擎的工作原理有了更深刻的理解。搜索引擎通过 PageRank 算法计算网页的重要性排名，将最有可能与用户查询相关的网页排在前面。这使得用户能够更快速地找到自己需要的信息。

当我了解 PageRank 算法时，我开始认识到矩阵运算等基本原理也可以有重大的应用。这个算法依赖于矩阵的乘法和特征向量的计算，这让我意识到，即使基本的数学原理看起来很简单，但它们在实际中的应用是极其广泛和重要的。不仅仅是 PageRank 算法，例如，在机器学习和深度学习中，矩阵运算也是基本的工具之一。神经网络中的权重矩阵，卷积神经网络中的卷积操作以及循环神经网络中的隐藏状态矩阵都是例子。这些工具被广泛用于图像识别，自然语言处理，推荐系统等等领域。

我开始认识到，基本的数学原理对于人工智能、机器学习、高性能计算、科学和工程等领域的发展都是至关重要的。在未来的学习和职业生涯中，我将更加深入地学习和应用这些基本的数学原理，以便更好地理解和创新在这些领域的技术和应用。

参考文献

[1] Search Engine Market Share Worldwide, Statcounter GlobalStats, 2023, https://gs.statcounter.com/search-engine-market-share

[2] Brin, S., Page, L.. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems.,1998, 30(1–7): 107–117.

[3] Rajeev Motwani, Prabhakar Raghavan. Randomized Algorithms. Cambridge University Press, United Kingdom, 1995.

[4] Avrachenkov K, Litvak N. Decomposition of the google pagerank and optimal linking strategy[D]. INRIA, 2004.

[5] 吴淑燕, 许涛. PageRank 算法的原理简介 [J]. 图书情报工作, 2003, 47(2): 55.

[6] 傅懷慧, 林共進, 白峰杉, 等. 阻尼因子對網頁排名之敏感度分析 [J]. 中國統計學報, 2005, 43(2): 145-164.

[7] Haveliwala T H. Topic-sensitive pagerank[C]. Proceedings of the 11th international conference on World Wide Web. 2002: 517-526.

[8] Gruetzmacher, T.; Cojean, T.; Flegar, G.; Anzt, H.; Quintana-Orti, ES. (2020). Acceleration of PageRank with customized precision based on mantissa segmentation. ACM Transactions on Parallel Computing. 7(1):1-19. https://doi.org/10.1145/3380934

[9] 顾亦然, 许梦馨. 基于 PageRank 的新闻关键词提取算法 [J]. 电子科技大学学报,2017(5):777-783.

[10] 李荣军, 王小捷, 周延泉. PageRank 模型在中文情感词极性判别中的应用 [J]. 北京邮电大学学报, 2010, 33(5): 141.

附 录

小组成员贡献:		
	武泽恺 10225101429	**25**%
	武泽恺 10225101429	**25**%
	武泽恺 10225101429	**25**%
	武泽恺 10225101429	**25**%