暨南大学本科实验报告专用纸

课程名称 数值计算实验 成绩评定
实验项目名称 Computing Problems 指导老师 Liangda Fang
实验项目编号 04 实验项目类型 验证型 实验地点
学生姓名 学号
学院 国际学院 系 专业 计算机科学与技术
实验时间 2023 年 12 月 1 日上午 10:30 ～ 12:10

I. Problem

1. Implement a simple web crawler;
2. Acquire the google matrix of the first 500 web pages from the main webpage of any university via the above web crawler, and give their adjacency matrix;
3. Implement the Power Method;
4. Compute the dominant eigenvector of the google matrix;
5. List the top 20 web pages.

II. Algorithm Summary

1. PageRank Algorithm

The PageRank algorithm, devised by Larry Page and Sergey Brin, co-founders of Google, is a method used to determine the importance of a webpage based on its relevance and connectivity within the web. It serves as a fundamental algorithm for Google's search engine, aiding in the sorting and ranking of webpages.

Key concepts of the PageRank algorithm include:

1. Link Graph: The internet is visualized as a vast graph where webpages are represented as nodes, and hyperlinks between them form directed edges.
2. Importance Transmission: PageRank measures the importance of a webpage by evaluating both the quantity and quality of incoming links. A webpage is considered more important if it is linked by other pages, especially those of high importance.
3. Markov Process: PageRank employs the concept of a Markov chain to model the transition between webpages. Webpages are treated as states, and the hyperlinks between them represent the probabilities of transitioning between states.
4. Iterative Calculation: The algorithm uses iterative methods to compute the weight (importance) of each webpage. Initially, all webpages are assigned equal weight, and through multiple iterations based on linking relationships, the weight of each webpage is updated until convergence.

5. Stability and Convergence: PageRank continues its iterative calculations to update webpage weights until they reach a stable state where further iterations do not significantly change the weights.

In short, the algorithm is implemented based on Markov process and power iteration method. Let's look at the following example flow:

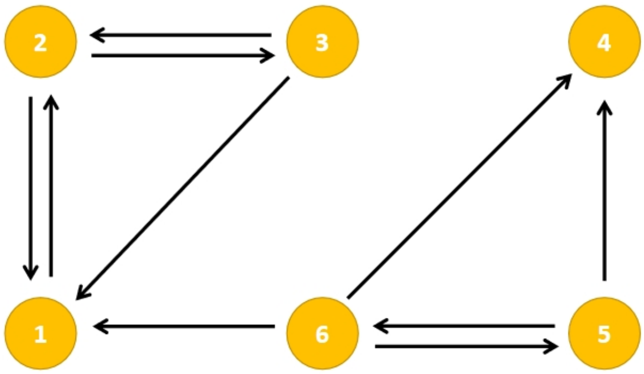


Figure 1: Page links diagram

This is a directed graph where outgoing links from a website refer to other pages. In this case, Site 1 refers to both Site 2 and Site 3. In computer science, directed graphs are commonly represented using an adjacency matrix $\mathbf{A}$. For instance:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Now, The adjacency matrix $\mathbf{A}$, we set $\mathbf{A_{ij} = 1}$ which indicates that Site i refers to Site j, while $\mathbf{A_{ij} = 0}$ signifies no reference. This representation helps depict the directed connections within the graph of web pages.

Typically, web pages are assigned priorities, and displaying high-priority pages at the top aims to facilitate easier access for users. However, determining the importance of these pages poses a challenge. We primarily rely on the relevance between web pages as a guide. If a webpage is linked to by many other pages, it implies higher importance. Similarly, pages referenced by high-priority pages might also carry significance. Common intuition suggests that users tend to spend more time on pages perceived as highly important.

Suppose there's a student named **Hyq** currently browsing the internet. **Hyq** has two choices while surfing:

1. Moving to a Random Site: **Hyq** might decide to move to a random site across the internet with a probability denoted as α.
2. Clicking a Link on the Current Page: Alternatively, **Hyq** may choose to click on a link present on the current page they're visiting with a probability of (1 - α).

It's assumed that the probability of **Hyq** clicking on a link on any given site is equal, and there exist 'n' total websites available. Consequently, the probability of **Hyq** transitioning from Site i to Site j can be expressed as:

$$p_{ji} = \begin{cases} \frac{\alpha}{n} + (1-\alpha) * \frac{A_{ji}}{\sum_{k=1}^{n} A_{ki}}, & \text{other} \\ \frac{1}{n}, & \sum_{k=1}^{n} A_{ki} = 0 \end{cases}$$

p_{ji} means Rosen goes from site i to j .

By this time, let's transform our adjacency matrix **A** into a "surfing" matrix **M**:

$$\mathbf{M} = \begin{bmatrix} \frac{a}{6} & \frac{a}{6} + \frac{(1-a)}{2} & \frac{a}{6} + \frac{(1-a)}{2} & \frac{1}{6} & \frac{a}{6} & \frac{a}{6} + \frac{(1-a)}{3} \\ \frac{a}{6} + (1-a) & \frac{a}{6} & \frac{a}{6} + \frac{(1-a)}{2} & \frac{1}{6} & \frac{a}{6} & \frac{a}{6} \\ \frac{a}{6} & \frac{a}{6} + \frac{(1-a)}{2} & \frac{a}{6} & \frac{1}{6} & \frac{a}{6} & \frac{a}{6} \\ \frac{a}{6} & \frac{a}{6} & \frac{a}{6} & \frac{1}{6} & \frac{a}{6} + \frac{(1-a)}{2} & \frac{a}{6} + \frac{(1-a)}{3} \\ \frac{a}{6} & \frac{a}{6} & \frac{a}{6} & \frac{1}{6} & \frac{a}{6} & \frac{a}{6} + \frac{(1-a)}{3} \\ \frac{a}{6} & \frac{a}{6} & \frac{a}{6} & \frac{1}{6} & \frac{a}{6} + \frac{(1-a)}{2} & \frac{a}{6} \end{bmatrix}$$

The **M** refers to the "Google matrix/Stochastic Matrix/Markov Matrix." An interesting fact to note is that all elements of the matrix are positive, and the sum of each column of the matrix equals 1. In other words, there exists a left eigenvector $\mathbf{v}$ = (1, 1, ⋯, 1):

$$\mathbf{vM} = \mathbf{v}$$

That means the matrix **M** has an eigenvalue of 1. As per the Gershgorin circle theorem, it's established that the eigenvalues of **M** cannot exceed 1. Given the positivity of **M**, according to the Perron-Frobenius theorem, it follows that the absolute values of all eigenvalues of the matrix are less than 1, excluding the aforementioned eigenvalue of 1.

Now, when we say matrix **M** possesses a steady-state distribution, what does it signify? Let's assume a vector **p**, and $\mathbf{p_i}$, represents the number of people staying on page i. Assuming:

$$\mathbf{P} = (100, 0, 0, ..., 0)$$

This implies that initially, there were 100 people staying on page 1. After a considerable duration, we're interested in determining the number of people currently staying on the various web pages:

$$\mathbf{P_{now}} = \lim_{n \to \infty} \mathbf{G}^n \mathbf{P}$$

You'll be amazed to observe the convergence of the matrix power $\mathbf{P_{now}}$! This incredible outcome is due to the fact that 1 is the largest absolute eigenvalue of **M** ! Take a look at this:

Assume $c_1, c_2 \neq 0$ and v_i is eigenvector of $\mathbf{M}$

$$\text{And } |\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$$

$$\mathbf{P} = c_1 v_1 + \cdots + c_n v_n$$

$$\mathbf{P_k} = \mathbf{M}^k \mathbf{P} = c_1 \lambda_1^k v_1 + \cdots + c_n \lambda_n^k v_n$$

$$\frac{\mathbf{P_k}}{\lambda_1^k} = c_1 v_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k v_n$$

$$\lim k \to \infty, \left(\frac{\lambda_j}{\lambda_1}\right)^k \to 0$$

That's precisely why $\mathbf{P_{now}}$ will converge! It will gradually converge towards the eigenvector of the greatest absolute eigenvalue of $\mathbf{M}$, eventually reaching a multiple of that eigenvalue. This process is referred to as the power iteration method.

By sorting the vector $\mathbf{P_{now}}$ from largest to smallest, we've successfully obtained the ranking of pages! We've accomplished it gracefully.

III. Experimental procedures

Step1: Define the Web_Crawler class to implement the method of crawling pages and building an adjacency matrix between pages.
Step2: Start crawling pages from "https://www.jnu.edu.cn" and continue crawling until 500 web pages have been crawled.
Step3: Draw the adjacency matrix of these pages and convert the matrix to a google matrix.
Step4: Implement the Power Iteration Method and calculate the dominant eigenvector of the google matrix.
Step5: List the top 20 web pages with the highest rank value.

IV. Result Analysis

First we crawled 500 web pages:

```
E:\Miniconda3\python.exe D:\iCloudDrive\Documents\文稿\JNU_Course\Junior\NumericalComputation\Lab\Lab4\H3Art\main.py
Crawling Page 1 : https://www.jnu.edu.cn , Status: 200
Crawling Page 2 : https://info.jnu.edu.cn , Status: 200
Crawling Page 3 : https://mail.jnu.edu.cn , Status: No-HTML content
Crawling Page 4 : https://ehall.jnu.edu.cn , Status: 200
Crawling Page 5 : https://lib.jnu.edu.cn , Status: 200
Crawling Page 6 : https://english.jnu.edu.cn , Status: 200
Crawling Page 7 : https://dsh.jnu.edu.cn , Status: 200
Crawling Page 8 : https://panyu.jnu.edu.cn , Status: 200
Crawling Page 9 : https://zh.jnu.edu.cn , Status: 200
Crawling Page 10 : https://sz.jnu.edu.cn , Status: 200
Crawling Page 11 : https://zsb.jnu.edu.cn , Status: 200
Crawling Page 12 : https://yz.jnu.edu.cn , Status: 200
Crawling Page 13 : https://hwy.jnu.edu.cn/13405/main.htm , Status: 200
Crawling Page 14 : https://jyxy.jnu.edu.cn , Status: 200
Crawling Page 15 : https://career.jnu.edu.cn , Status: 200
Crawling Page 16 : https://faculty.jnu.edu.cn , Status: 200
Crawling Page 17 : https://jwc.jnu.edu.cn , Status: 200
Crawling Page 18 : https://hwy.jnu.edu.cn , Status: 200
Crawling Page 19 : https://kjc.jnu.edu.cn , Status: 200
Crawling Page 20 : https://skc.jnu.edu.cn , Status: 200
Crawling Page 21 : https://www.jnu.edu.cn/jxky/list.htm , Status: 200
Crawling Page 22 : https://wsc.jnu.edu.cn , Status: 200
Crawling Page 23 : https://gccgl.jnu.edu.cn/jnuzpf/a/zp/base/Index , Status: 200
Crawling Page 24 : https://hrdam.jnu.edu.cn/343/list.htm , Status: 200
Crawling Page 25 : https://zhaopin.jnu.edu.cn , Status: 200
Crawling Page 26 : https://zwc.jnu.edu.cn , Status: 200
Crawling Page 27 : https://netc.jnu.edu.cn , Status: 200
Crawling Page 28 : https://www.jnu.edu.cn/tz/list.psp , Status: 200
Crawling Page 29 : https://www.jnu.edu.cn/gg/list.psp , Status: 200
Crawling Page 30 : https://news.jnu.edu.cn/jnyw/yw/2023/11/18/18371452545.html , Status: 404
Crawling Page 31 : https://mp.weixin.qq.com/s/sfhARIabj97V1A7Wc1JPQg , Status: 200
Crawling Page 32 : https://news.jnu.edu.cn/col2.html , Status: 200
Crawling Page 33 : https://ddesd.jnu.edu.cn , Status: 200
Crawling Page 34 : https://news.jnu.edu.cn/col11.html , Status: 200
Crawling Page 35 : https://news.jnu.edu.cn/col3.html , Status: 200
Crawling Page 36 : https://news.jnu.edu.cn , Status: 200
Crawling Page 37 : https://jwc.jnu.edu.cn/2023/1201/c8320a783087/page.htm , Status: 200
Crawling Page 38 : https://jwc.jnu.edu.cn/2023/0823/c8320a761121/page.htm , Status: 200
Crawling Page 39 : https://news.jnu.edu.cn/col8.html , Status: 200
Crawling Page 40 : https://news.jnu.edu.cn/col9.html , Status: 200
Crawling Page 41 : https://news.jnu.edu.cn/col5.html , Status: 200
```

Figure 2: Crawling websites

```
Crawling Page 468 : https://hwy.jnu.edu.cn/2022/0624/c13398a707387/page.htm , Status: 200
Crawling Page 469 : https://hwy.jnu.edu.cn/13399/list.htm , Status: 200
Crawling Page 470 : https://hwy.jnu.edu.cn/2022/0622/c13403a706495/page.htm , Status: 200
Crawling Page 471 : https://hwy.jnu.edu.cn/13397/list.htm , Status: 200
Crawling Page 472 : https://hwy.jnu.edu.cn/13402/list.htm , Status: 200
Crawling Page 473 : https://hwy.jnu.edu.cn/2023/0831/c13397a761843/page.htm , Status: 200
Crawling Page 474 : https://hwy.jnu.edu.cn/xsjy/main.htm , Status: 200
Crawling Page 475 : https://ehall.jnu.edu.cn/infoplus/form/KYXMYSTZSQ/start , Status: 200
Crawling Page 476 : https://kjc.jnu.edu.cn/2016/1206/c3501a93205/page.htm , Status: 200
Crawling Page 477 : https://kjc.jnu.edu.cn/_web/_onlinesurvey/api/lookOnlineSurvey/new.rst?_p=YXM9NTImdD0xMTEmcD0xJm09TiY_&banner=1&tt=0.8846275260812899&request_locale=zh_CN&localeCH=true&id=25&check=2 , Status: 200
Crawling Page 478 : https://kjc.jnu.edu.cn/2021/0705/c917a635937/page.htm , Status: 200
Crawling Page 479 : https://cwc.jnu.edu.cn/f1/31/c4234a258353/page.psp , Status: 200
Crawling Page 480 : https://cwc.jnu.edu.cn , Status: 200
Crawling Page 481 : https://hrdam.jnu.edu.cn , Status: 200
Crawling Page 482 : https://kjc.jnu.edu.cn/33611/list.htm , Status: 200
Crawling Page 483 : https://kjc.jnu.edu.cn/2023/1207/c34033a783665/page.htm , Status: 200
Crawling Page 484 : https://kjc.jnu.edu.cn/33211/list.htm , Status: 200
Crawling Page 485 : https://kjc.jnu.edu.cn/2023/1122/c34033a782055/page.htm , Status: 200
Crawling Page 486 : https://kjc.jnu.edu.cn/33205/list.htm , Status: 200
Crawling Page 487 : https://kjc.jnu.edu.cn/2023/1205/c34033a783403/page.htm , Status: 200
Crawling Page 488 : https://kjc.jnu.edu.cn/33645/list.htm , Status: 200
Crawling Page 489 : https://kjc.jnu.edu.cn/33193/list.htm , Status: 200
Crawling Page 490 : https://kjc.jnu.edu.cn/33607/list.htm , Status: 200
Crawling Page 491 : https://kjc.jnu.edu.cn/33219/list.htm , Status: 200
Crawling Page 492 : https://kjc.jnu.edu.cn/2022/0516/c33581a699277/page.htm , Status: 200
Crawling Page 493 : https://kjc.jnu.edu.cn/784/list.htm , Status: 200
Crawling Page 494 : https://kjc.jnu.edu.cn/33215/list.htm , Status: 200
Crawling Page 495 : https://kjc.jnu.edu.cn/33625/list.htm , Status: 200
Crawling Page 496 : https://kjc.jnu.edu.cn/33173/list.htm , Status: 200
Crawling Page 497 : https://kjc.jnu.edu.cn/33603/list.htm , Status: 200
Crawling Page 498 : https://kjc.jnu.edu.cn/33207/list.htm , Status: 200
Crawling Page 499 : https://skc.jnu.edu.cn/9317/list.htm , Status: 200
Crawling Page 500 : https://skc.jnu.edu.cn/9340/list.htm , Status: 200
Crawling Page 501 : https://skc.jnu.edu.cn/9355/list.htm , Status: 200
Crawling Page 502 : https://xsc.jnu.edu.cn/5840/list.htm , Status: 200
Crawling Page 503 : https://www.jnu.edu.cn/2022/0829/c2593a714477/page.htm , Status: 200
Crawling Page 504 : https://english.jnu.edu.cn/2016/1031/c2490a47517/page.htm , Status: No-HTML content
Crawling Page 505 : https://live.southcn.com , Status: 200
Crawling Page 506 : https://jyxy.jnu.edu.cn/28101/list.htm , Status: 200
Crawling Page 507 : https://gddj.southcn.com , Status: 200
Crawling completed. Total sites: 500
keys: 500
```

Figure 3: Crawling websites(cont.)

Based on the link relationships between these 500 pages, an adjacency matrix diagram can be constructed as follows:

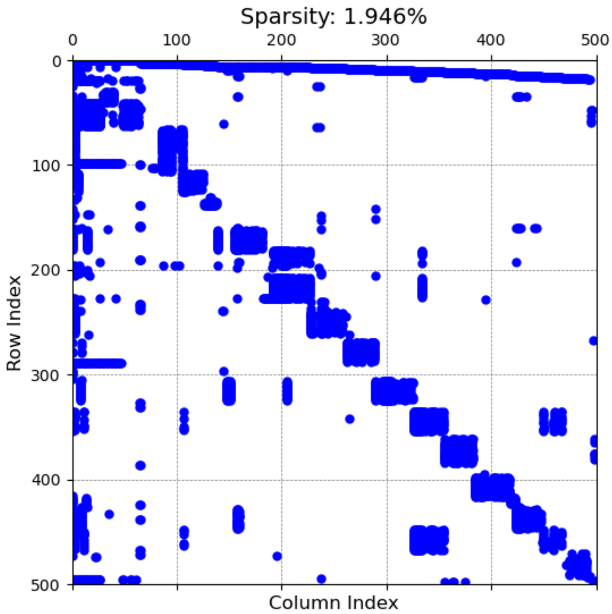
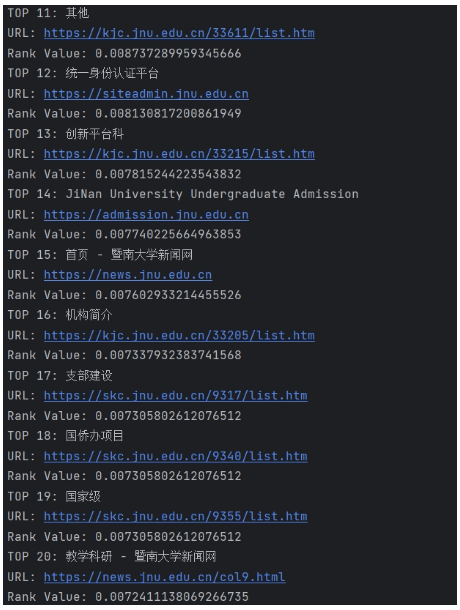


Figure 4: Sparsity matrix of these 500 pages

According to the PageRank algorithm, we get the top 20 web pages with the highest rank values:

```
TOP 1: 暨南大学图书馆
URL: https://lib.jnu.edu.cn
Rank Value: 0.036824973012285524
TOP 2: 暨南大学-华侨最高学府-国家“双一流”建设高校
URL: https://www.jnu.edu.cn
Rank Value: 0.03621698036050209
TOP 3: 暨南大学图书馆
URL: https://lib.jnu.edu.cn/department/index/11
Rank Value: 0.0240492226913659
TOP 4: 图书馆概况-暨南大学图书馆
URL: https://lib.jnu.edu.cn/home/service/0/2774
Rank Value: 0.020536069120696315
TOP 5: 暨南大学统一身份认证登录指南
URL: https://netc.jnu.edu.cn/2020/1124/c10374a565499/page.htm
Rank Value: 0.02033873219076459
TOP 6: Mynet校园网用户自助服务平台
URL: https://mynet.jnu.edu.cn/app/forgetPassword.html
Rank Value: 0.02033873219076459
TOP 7: 暨南大学图书馆
URL: https://lib.jnu.edu.cn/department/index/36
Rank Value: 0.01763508439933019
TOP 8: 暨南大学图书馆
URL: https://lib.jnu.edu.cn/department/index/22
Rank Value: 0.017047205259862395
TOP 9: 暨南大学图书馆
URL: https://lib.jnu.edu.cn/department/index/37
Rank Value: 0.01685197034276468
TOP 10: 校地合作
URL: https://kjc.jnu.edu.cn/33625/list.htm
Rank Value: 0.009215398454833004
```

Figure 5: Top 20 web pages(1-10)



```
TOP 11: 其他
URL: https://kjc.jnu.edu.cn/33611/list.htm
Rank Value: 0.008737289959345666
TOP 12: 统一身份认证平台
URL: https://siteadmin.jnu.edu.cn
Rank Value: 0.008130817200861949
TOP 13: 创新平台科
URL: https://kjc.jnu.edu.cn/33215/list.htm
Rank Value: 0.007815244223543832
TOP 14: JiNan University Undergraduate Admission
URL: https://admission.jnu.edu.cn
Rank Value: 0.007740225664963853
TOP 15: 首页 - 暨南大学新闻网
URL: https://news.jnu.edu.cn
Rank Value: 0.007602933214455526
TOP 16: 机构简介
URL: https://kjc.jnu.edu.cn/33205/list.htm
Rank Value: 0.007337932383741568
TOP 17: 支部建设
URL: https://skc.jnu.edu.cn/9317/list.htm
Rank Value: 0.007305802612076512
TOP 18: 国侨办项目
URL: https://skc.jnu.edu.cn/9340/list.htm
Rank Value: 0.007305802612076512
TOP 19: 国家级
URL: https://skc.jnu.edu.cn/9355/list.htm
Rank Value: 0.007305802612076512
TOP 20: 教学科研 - 暨南大学新闻网
URL: https://news.jnu.edu.cn/col9.html
Rank Value: 0.0072411138069266735
```

Figure 6: Top 20 web pages(11-20)

Finally, we can plot the website rank value distribution:

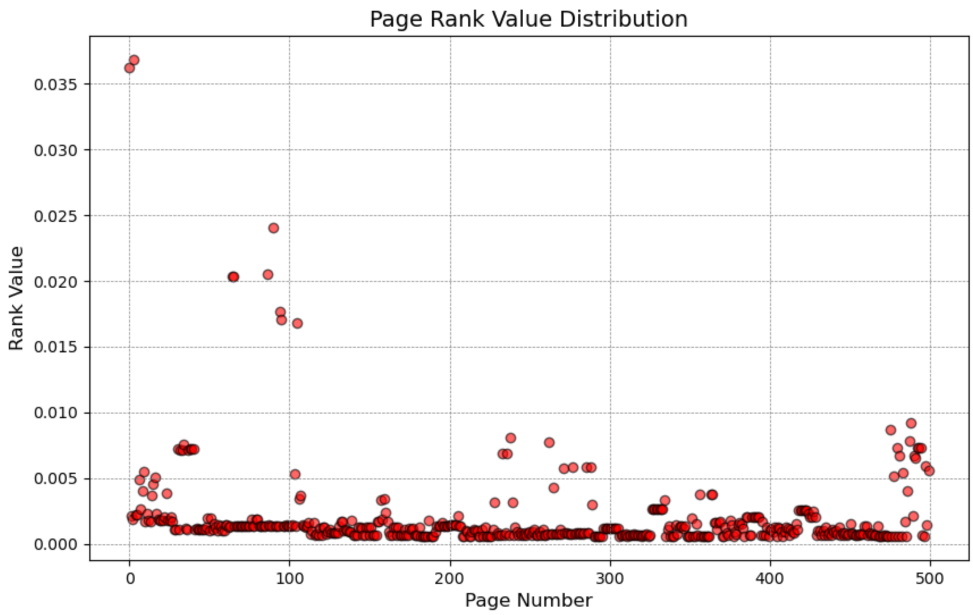


Figure 7: Rank value distribution

V. Experimental Summary

In this lab, we extracted data from 500 distinct web pages linked to Jinan University's official site, "https://www.jnu.edu.cn", forming an adjacency matrix **A**. Utilizing **A**, we constructed the Google matrix **G** and employed the power iteration algorithm to determine the relative importance and rankings of these sites.

It's important to note that for the adjacency matrix **A**, we ensured the sum of each column equals 1, contrary to the row sum. Additionally, we imposed a limit on the number of relative URLs per page to a maximum of 20. This was particularly relevant for pages like "https://www.jnu.edu.cn" to prevent overrepresentation of nodes from the same website's sub-pages. Moreover, we encountered numerous URLs referring to the same site (for instance, "http://www.jnu.edu.cn", "https://www.jnu.edu.cn", and "https://www.jnu.edu.cn/"). To address this, we focused exclusively on URLs starting with 'https' and eliminated any trailing '/' to avoid redundancy.

VI. Appendix: Source Code

In this experiment, the code implementation part uses Python and a series of related third-party libraries, including: **numpy**, **matplotlib**, **requests** and **beautifulsoup4**.

1. web_crawler.py

```python
from requests import get, head
from bs4 import BeautifulSoup
from random import sample
from urllib.parse import urljoin

# Constants for crawler configuration
MAX_SITES = 500
MAX_RELATIVE_LINKS = 20


class Web_Crawler:
    HEADERS = {
        'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36'
    }

    def __init__(self, base_url, verbose=False):
        self.to_crawl = [base_url]
        self.site_count = 1
        self.crawled = set()
        self.unreachable = set()
        self.link_graph = {}
        self.verbose = verbose

    def extract_links(self, url):
        try:
            response = head(url)
            if 'text/html' not in response.headers.get('Content-Type', ''):
                raise ValueError("No-HTML content")
            response = get(url, headers=self.HEADERS, timeout=10)
        except:
            self.crawled.remove(url)
            self.unreachable.add(url)
            self.site_count = self.site_count - 1
            return -1

        response.encoding = 'utf-8'
        soup = BeautifulSoup(response.text, 'lxml')

        # Collect and normalize links
        links = [link['href'] for link in soup.find_all('a', href=True)]
        links = self._normalize_links(url, links)

        for link in links:
            if link not in self.to_crawl and \
                    link not in self.crawled and \
                    link not in self.unreachable and \
                    self.site_count < MAX_SITES:
                self.to_crawl.append(link)
                self.site_count += 1
```

```python

        self.link_graph[url] = links
        response.close()
        return response.status_code

    def _normalize_links(self, base_url, links):
        relative = [link for link in links if link.startswith('/')]
        absolute = [link for link in links if link.startswith('https')]
        relative = sample(relative, min(len(relative), MAX_RELATIVE_LINKS))
        links = [link.rstrip('/') for link in absolute + [urljoin(base_url, rel) for rel in relative]]
        return links

    def run(self):
        index = 1
        while self.to_crawl:
            current_url = self.to_crawl.pop(0)
            self.crawled.add(current_url)
            status = self.extract_links(current_url)
            if self.verbose:
                print('Crawling Page', index, ':', current_url, ', Status:',
                      'No-HTML content' if status == -1 else status)
                index += 1
        if self.verbose:
            print('Crawling completed. Total sites:', self.site_count)
            print('keys:', len(self.link_graph.keys()))

    def get_adjacent_matrix(self):
        keys = tuple(self.link_graph.keys())

        for key in keys:
            for site in self.link_graph[key]:
                if site not in keys or site == key:
                    self.link_graph[key] = [x for x in self.link_graph[key] if x != site]

        matrix = [[0] * self.site_count for _ in range(self.site_count)]

        for site, links in self.link_graph.items():
            site_index = keys.index(site)
            for link in links:
                link_index = keys.index(link)
                matrix[site_index][link_index] = 1

        return matrix


def get_page_title(url):
    try:
        response = get(url, headers=Web_Crawler.HEADERS)
        response.encoding = 'utf-8'
        soup = BeautifulSoup(response.text, 'lxml')
        title = soup.find('title').text
    finally:
        response.close()
    return title
```

2. algorithm.py

```
import numpy as np


def power_iteration(matrix):
    num_nodes = matrix.shape[0]
    # Initialize vector: [1, 0, 0, ..., 0]
    rank_vector = np.zeros(num_nodes)
    rank_vector[0] = 1
    while True:
        new_rank_vector = np.dot(matrix, rank_vector)
        change = np.linalg.norm(new_rank_vector - rank_vector)
        # Convergence check (threshold set arbitrarily)
        if change < 1e-20:
            break
        else:
            rank_vector = new_rank_vector

    return new_rank_vector


def convert_to_google_matrix(adj_matrix, num_sites, alpha):
    for row in range(num_sites):
        row_sum = sum(adj_matrix[row])
        if row_sum == 0:
            for col in range(num_sites):
                adj_matrix[row][col] = 1 / num_sites
        else:
            for col in range(num_sites):
                random_surf = alpha * (1 / num_sites)
                outlink_surf = (1 - alpha) * (adj_matrix[row][col] / row_sum)
                adj_matrix[row][col] = random_surf + outlink_surf
    return adj_matrix
```

3. utils.py

```python
import matplotlib.pyplot as plt
import numpy as np
from json import dump, load

# Constants
DATA_FILE = 'crawled_data.json'
ALPHA = 0.15
NUM_TOP_PAGES = 20


def save_crawl_data(crawler, filename):
    data = {
        'adjacent_matrix': crawler.get_adjacent_matrix(),
        'directed_graph': crawler.link_graph,
        'num_site': crawler.site_count
    }
    with open(filename, 'w', encoding='utf-8') as file:
        dump(data, file, ensure_ascii=False, indent=4)


def load_data(filename):
    with open(filename, 'r') as file:
        return load(file)


def plot_sparsity(matrix):
    size = len(matrix)
    plt.figure(figsize=(10, 6))  # Set a larger figure size
    plt.spy(matrix, markersize=5, marker='o', color='blue')  # Increase marker size and change color
    plt.xlim([0, size])
    plt.ylim([size, 0])

    # Calculate sparsity
    non_zero_count = np.count_nonzero(matrix)
    sparsity_percentage = (non_zero_count / (size ** 2)) * 100
    plt.title(f'Sparsity: {sparsity_percentage:.3f}%', fontsize=14)
    plt.xlabel('Column Index', fontsize=12)
    plt.ylabel('Row Index', fontsize=12)
    plt.grid(color='gray', linestyle='--', linewidth=0.5)
    plt.show()


def plot_rank_distribution(rank_values):
    plt.figure(figsize=(10, 6))  # Set a larger figure size
    plt.scatter(range(len(rank_values)), rank_values, color='red', alpha=0.6,
                edgecolor='black')  # Adjust color, transparency, and edge
    plt.title('Page Rank Value Distribution', fontsize=14)
    plt.xlabel('Page Number', fontsize=12)
    plt.ylabel('Rank Value', fontsize=12)
    plt.grid(True, color='gray', linestyle='--', linewidth=0.5)
    plt.show()
```

4. main.py

```python
from web_crawler import Web_Crawler, get_page_title
```

```python
from algorithm import power_iteration, convert_to_google_matrix
from utils import *
from os.path import exists
from datetime import datetime
import numpy as np


def crawl_website(base_url, verbose=True):
    crawler = Web_Crawler(base_url=base_url, verbose=verbose)
    start_time = datetime.now()
    crawler.run()
    end_time = datetime.now()
    print('Crawling time used:', (end_time - start_time).total_seconds(), 'seconds')
    return crawler


def display_top_pages(page_rank, num_pages):
    for i, (url, rank) in enumerate(page_rank[:num_pages]):
        title = get_page_title(url)
        print(f'TOP {i + 1}: {title}\nURL: {url}\nRank Value: {rank}')


def main():
    # Crawl and save data if not already done
    if not exists(DATA_FILE):
        crawler = crawl_website('https://www.jnu.edu.cn')
        save_crawl_data(crawler, DATA_FILE)

    # Load data from file
    data = load_data(DATA_FILE)

    # Plotting and processing
    plot_sparsity(data['adjacent_matrix'])
    google_matrix = convert_to_google_matrix(np.array(data['adjacent_matrix'], dtype=float), data['num_site'], ALPHA)

    # Power iteration
    rank_values = power_iteration(google_matrix.transpose())
    plot_rank_distribution(rank_values)

    # Ranking and displaying top pages
    ranked_pages = [(key, rank_values[i]) for i, key in enumerate(data['directed_graph'].keys())]
    page_rank = sorted(ranked_pages, key=lambda x: x[1], reverse=True)
    display_top_pages(page_rank, NUM_TOP_PAGES)


if __name__ == "__main__":
    main()
```