STA 141B Assignment 3

Due **XXXXX, 2024** by **11:59pm**. Submit your work by uploading it to Gradescope through Canvas.

Instructions:

1. Provide your solutions in new cells following each exercise description. Create as many new cells as necessary. Use code cells for your Python scripts and Markdown cells for explanatory text or answers to non-coding questions. Answer all textual questions in complete sentences.
2. The use of assistive tools is permitted, but must be indicated. You will be graded on you proficiency in coding. Produce high quality code by adhering to proper programming principles.
3. Export the .jpynb as .pdf and submit it on Gradescope in time. To facilitate grading, indicate the area of the solution on the submission. Submissions without indication will be marked down. No late submissions accepted.
4. If test cases are given, your solution must be in the same format.
5. The total number of points is 10.

Exercise 1

We will compute the PageRank of the articles of the Hawaiian wikipedia, which is available at haw.wikipedia.org. Additional information of the Hawaiian wiki can be found here.

Hints: If you don't speak Hawaiian, you might want to learn the wiki logic from the English wikipedia, and translate your findings. Also, caching is recommended.

(a) Use the special AllPages page and understand its logic to retrieve the url of all articles in the Hawaiian wikipedia. Make sure to skip redirections.

How many articles did you find? (I found a bit more than 2541.)

In [23]:
```
# a)
import requests
import requests_cache
import lxml.html as lx
import re
```

In [74]:
```
requests_cache.install_cache("hawaii_wiki_cache")
curr_link = "https://haw.wikipedia.org/wiki/Papa_nui:AllPages"
response = requests.get(curr_link)
page = lx.fromstring(response.text)
next_page = 1
```

```
visited_links = {}
next_pages = set()
while next_page not in next_pages:
    next_pages.add(next_page)
    response = requests.get(curr_link)
    page = lx.fromstring(response.text)
    next_page = page.xpath('//*[@id="mw-content-text"]/div[2]/a[last()]/@hre
    next_page = f'https://haw.wikipedia.org{next_page}'
    links = page.xpath('//*[@id="mw-content-text"]/div[3]/ul/li[not(contains
    for link in links:
        if link.startswith('/wiki'):
            link = f'https://haw.wikipedia.org{link}'
        visited_links[link] = []
    curr_link = next_page
len(visited_links)
```

Out[74]: 2629

ANSWER

I got 2629 total links from the allpages

(b, i) Write a function that scans an article given by its url and retrieves all links to other articles in the Hawaiian wikipedia. Avoid links to special pages, images or the ones that point to another website. Only count the proper article for links that point to a specific section. Use regular expressions to manage these cases. **(ii)** Make sure to match redirections to their correct destiation article. To this end, find how wikipedia treats redirections and retrieve the true article. *(Help: Try searching for 'uc davis' on en.wikipedia.org')* To this end, I used the collection or article urls obtained in (a), which I stored in a dict object to allow for fast lookups. Then, for each new found link I checked whether that link appeared in the dict. If not, It might be a re-direction and receive special attention.
(iii) Request all articles and obtain all links to other articles.

How many links to other articles are there? (I found 9,000-10,000.)

In [104…

```
requests_cache.install_cache("hawaii_wiki_cache")
def scan_article(urls):
    counter = 0
    article_matches = re.compile(r'^/wiki/[^:#]+$')
    for url in urls:
        response = requests.get(url)
        page = lx.fromstring(response.text)
        valid_links = page.xpath('//*[@id="mw-content-text"]//a[not(parent::
)
        redirects = page.xpath('//*[@id="mw-content-text"]//a[@class="mw-red
        for link in valid_links:
            link = f'https://haw.wikipedia.org{link}'
        for redirect in redirects:
            redirect = f'https://haw.wikipedia.org{redirect}'
```

```
            if redirect in urls:
                valid_links.append(redirect)
            else:
                response = requests.get(redirect)
                new_page = lx.fromstring(response.text)
                secondary_redirect = new_page.xpath('//*[@id="mw-content-tex
                for sec_redirect in secondary_redirect:
                    valid_links.append(sec_redirect)
        for link in valid_links:
            if article_matches.match(link):
                link  = f'https://haw.wikipedia.org{link}'
                urls[url].append(link)
        counter += len(valid_links)


    return counter, urls
```

In [105…
```
total, returned_links = scan_article(visited_links)
```

In [106…
```
total
```

Out[106… 9785

ANSWER

I got a total of 9785 links.

(c) Compute the transition matrix (see here and here for step-by-step instructions). Make sure to tread dangling nodes. You may want to use:

```
import numpy as np
from scipy.sparse import csr_matrix
```

In [107…
```
import numpy as np
from scipy.sparse import csr_matrix
link_keys = list(returned_links.keys())
link_indexing = {}
for i in range(len(link_keys)):
    link_indexing[link_keys[i]] = i
```

ANSWER

In [108…
```
transition_matrix = csr_matrix((len(link_keys), len(link_keys)))
for i in link_keys:
    outgoing_links = returned_links[i]
    if outgoing_links == []:
        transition_matrix[link_indexing[i], 0:-1] = 1/(len(link_keys))
    else:
```

```
        for outgoing in outgoing_links:
            transition_matrix[link_indexing[i], link_indexing[outgoing]] = 1
```

(d, i) Set the damping factor to `0.85` and compute the PageRank for each article, using fourty iterations and starting with a vector with equal entries. **(ii)** Obtain the top ten articles in terms of PageRank, and, retrieving the articles again, find the correponding English article, if available.

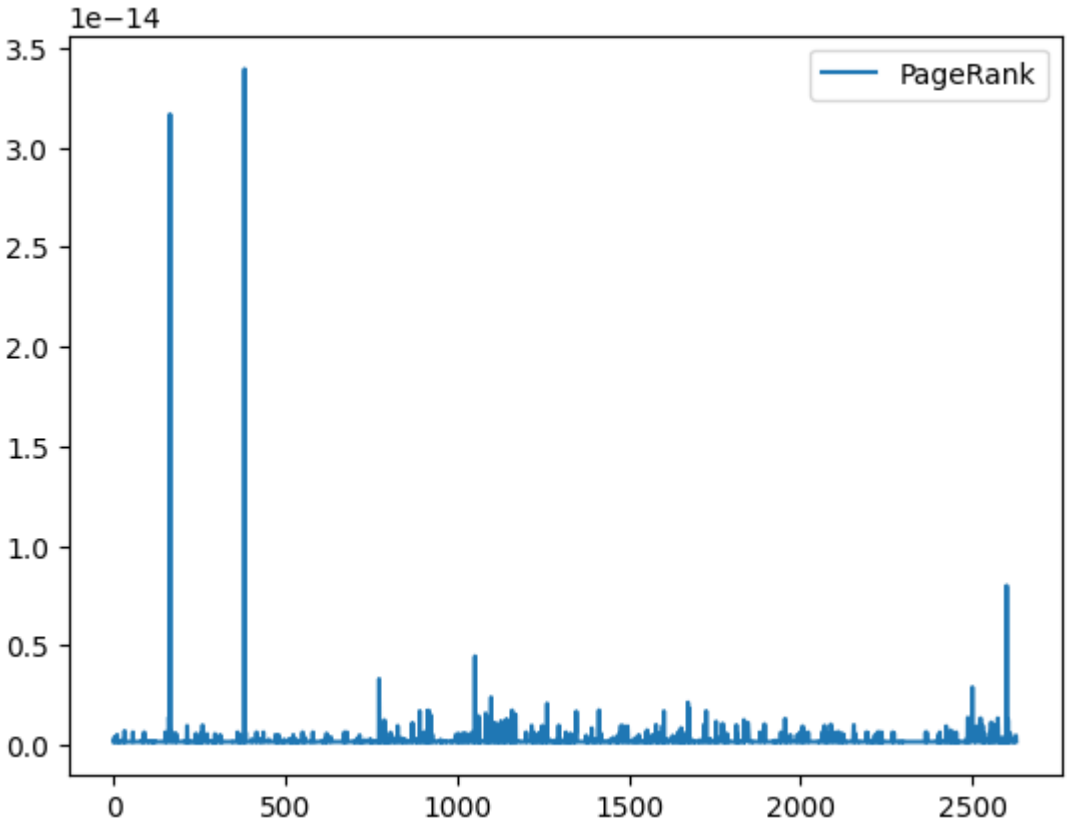
Return the corresponding English article titles of the top ten articles from the Hawaiian wikipedia.

In [109…
```
d = 0.85
equal_matrix = np.matrix(np.ones((len(link_keys), len(link_keys))))
equal_matrix = 1/len(link_keys) * equal_matrix
page_rank_matrix = (d * transition_matrix) + ((1-d) * equal_matrix)
equal_vector = (1/len(link_keys)) * np.ones((1, len(link_keys)))
```

In [110…
```
def page_rank_calc(vector, matrix, count = 1):
    while count != 40:
        vector = vector * matrix
        return page_rank_calc(vector, matrix, count + 1)
    return(vector * matrix)
rank_vector = page_rank_calc(equal_vector, page_rank_matrix).A1
```

ANSWER for d i)

In [111…
```
import matplotlib.pyplot as plt
x_val = np.arange(len(link_keys))
plt.plot(x_val, rank_vector, label = "PageRank")
plt.legend()
plt.show()
```



ANSWER for d ii)

In [112…
```
sorted_indices = np.argsort(rank_vector)[::-1]
top_indices = sorted_indices[:10]
top_articles = []
for key, value in link_indexing.items():
    if value in top_indices:
        top_articles.append(key)
top_articles
```

Out[112…
```
['https://haw.wikipedia.org/wiki/Aupuni_kiwik%C4%81',
 'https://haw.wikipedia.org/wiki/Castille_a_Leon',
 'https://haw.wikipedia.org/wiki/Hawai%CA%BBi',
 'https://haw.wikipedia.org/wiki/Kapikala',
 'https://haw.wikipedia.org/wiki/Kepania',
 'https://haw.wikipedia.org/wiki/Lituania',
 'https://haw.wikipedia.org/wiki/Palakila',
 'https://haw.wikipedia.org/wiki/Palani',
 'https://haw.wikipedia.org/wiki/%CA%BBAmelika_Hui_P%C5%AB_%CA%BBia',
 'https://haw.wikipedia.org/wiki/%CA%BB%C5%8Clelo_Pelekania']
```

In [113…
```
en_articles = []
for article in top_articles:
    new_link = article.replace('haw', 'en')
    response = requests.get(new_link)
    new_page = lx.fromstring(response.text)
    if not new_page.xpath('//*[@id="noarticletext"]'):
        en_articles.append(new_link)
```

```
en_articles
```

Out[113…
```
['https://en.wikipedia.org/wiki/Hawai%CA%BBi',
 'https://en.wikipedia.org/wiki/Lituania',
 'https://en.wikipedia.org/wiki/Palani']
```

In [15]:
```
import matplotlib.pyplot as plt
plt.plot(range(n), v, label = "PageRank")
plt.legend()
plt.show()
```