STA 220 Assignment 2

Due **Februrary 9, 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 Sinhala wikipedia, which is available at si.wikipedia.org. Additional information of the Sinhala wiki can be found here.

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

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

How many articles are there?

In [ ]:
```
from bs4 import BeautifulSoup
import requests
import re
import pickle
```

In [ ]:
```
#TEST CODE ON ENGLISH ALL PAGE
"""#starting url
url = "https://en.wikipedia.org/wiki/Special:AllPages"
#base wikipedia url
base = "https://en.wikipedia.org"
#get the response
response = requests.get(url)
bs = BeautifulSoup(response.text)
#hard code the next url


url_list = []
redirect_list = []
#First page
for link in bs.find('ul', {'class':'mw-allpages-chunk'}).find_all('a', {'href':re.c
    if link.get('class') == ['mw-redirect']:
        redirect_list.append(base + link.get('href'))
    else:
        full_url = base + link.get('href')
        url_list.append(full_url)
ref_list = bs.find('div', {'class':'mw-allpages-nav'}).find_all('a', {'href':re.com
if len(ref_list) == 1:
    next_url = base + ref_list[0].get('href')
else:
    next_url = base + ref_list[1].get('href')


#Begin while loop and stops when the navigate has only 1 element
response = requests.get(next_url)
ref_list = [0,1]
i = 0
while len(ref_list) > 1 and i < 5:
    bs = BeautifulSoup(response.text)
    ref_list = bs.find('div', {'class':'mw-allpages-nav'}).find_all('a', {'href':re
    if len(ref_list) == 1:
        next_url = base + ref_list[0].get('href')
    else:
        next_url = base + ref_list[1].get('href')
    for link in bs.find('ul', {'class':'mw-allpages-chunk'}).find_all('a', {'href':
        if link.get('class') == ['mw-redirect']:
            redirect_list.append(base + link.get('href'))
        else:
            full_url = base + link.get('href')
            url_list.append(full_url)
    i += 1 """
```

In [ ]:
```
#THIS IS THE CODE TO GET SINHALA URL FROM ALL PAGES
"""url = "https://si.wikipedia.org/wiki/%E0%B7%80%E0%B7%92%E0%B7%81%E0%B7%9A%E0%B7%
#base wikipedia url
base = "https://si.wikipedia.org"
#get the response
response = requests.get(url)
bs = BeautifulSoup(response.text)
#hard code the next url


url_list = []
redirect_list = []
#First page
for link in bs.find('ul', {'class':'mw-allpages-chunk'}).find_all('a', {'href':re.c
    if link.get('class') == ['mw-redirect']:
        redirect_list.append(base + link.get('href'))
    else:
        full_url = base + link.get('href')
        url_list.append(full_url)
ref_list = bs.find('div', {'class':'mw-allpages-nav'}).find_all('a', {'href':re.com
if len(ref_list) == 1:
    next_url = base + ref_list[0].get('href')
else:
    next_url = base + ref_list[1].get('href')


#Begin while loop and stops when the navigate has only 1 element
response = requests.get(next_url)
ref_list = [0,1]
i = 0
while len(ref_list) > 1:
    bs = BeautifulSoup(response.text)
    ref_list = bs.find('div', {'class':'mw-allpages-nav'}).find_all('a', {'href':re
    if len(ref_list) == 1:
        next_url = base + ref_list[0].get('href')
    else:
        next_url = base + ref_list[1].get('href')
    for link in bs.find('ul', {'class':'mw-allpages-chunk'}).find_all('a', {'href':
        if link.get('class') == ['mw-redirect']:
            redirect_list.append(base + link.get('href'))
        else:
            full_url = base + link.get('href')
            url_list.append(full_url)
    response = requests.get(next_url)"""
```

In [ ]:
```
#Save list of links to a txt file for easy access later
"""with open('../Data/links.txt', 'w') as f:
    for link in url_list:
        f.write(f"{link}\n")"""
```

In [ ]:
```
#Read URL List and save to list
with open('../Data/links.txt') as f:
    url_list = f.read().splitlines()
```

In [ ]:
```
len(url_list)
```

Out[ ]: 24234

(b, i) Scan all articles in the sinhalese wikipedia and retrieve all links to other articles. 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')* **(iii)** Use threading to request all articles and obtain all links to other articles. *(Attention: This takes about thirty minutes!)*

How many links to other articles are there?

In [ ]:
```
#import pickkle for easy dictionary save
import pickle
```

In [ ]:
```
#Create function that takes a URL and traverses through the article body and saves
def url_traversal(url):
    base = "https://si.wikipedia.org"
    url_dictionary = {}
    response = requests.get(url)
    bs = BeautifulSoup(response.text)
    summary = bs.find('div', {'class':'mw-content-ltr mw-parser-output'}).find_all(
    test_list = []
    if len(summary) > 1:
        for paragraph in summary[1:]:
            links_list = paragraph.find_all('a',{'href':re.compile("/w")})
            for link in links_list:
                if link.get('class') == ['new']:
                    continue
                elif link.get('class') == ['mw-redirect']:
                    redirect_url = base + link.get('href')
                    response_loop = requests.get(redirect_url).text
                    bs_loop = BeautifulSoup(response_loop)
                    test_list.append(bs_loop.find('link', {'rel':'canonical'}).get(
                else:
                    test_list.append(base + link.get('href'))
    elif len(summary) == 1:
        links_list = summary[0].find_all('a',{'href':re.compile("/w")})
        for link in links_list:
            if link.get('class') == ['mw-redirect']:
                redirect_url = base + link.get('href')
                response_loop = requests.get(redirect_url).text
                bs_loop = BeautifulSoup(response_loop)
                test_list.append(bs_loop.find('link', {'rel':'canonical'}).get('hre
            else:
                test_list.append(base + link.get('href'))
    else:
        pass
    url_dictionary[url] = test_list
    return url_dictionary
```

In [ ]:
```
#Use threads to speed up this process this definition takes list of url and goes th
"""import concurrent.futures
def threaded_traversal(url_dictionary, url_list):
    with concurrent.futures.ThreadPoolExecutor(max_workers = 16) as executor:
        for result in executor.map(url_traversal, url_list):
            url_dictionary.update(result)"""
```

In [ ]:
```
#create empty dictionary and save a dictionary of links to other articles
"""url_dictionary = {}
threaded_traversal(url_dictionary, url_list[0:20])"""
```

In [ ]:
```
#saves the dictionary into another file for ease of access in the future
#with open("../Data/url_dict.pkl", "ab") as f:
#    pickle.dump(url_dictionary, f)
```

In [ ]:
```
#read the dictionary file and save it into a dictionary
d = {}
with open("../Data/url_dict.pkl", "rb") as f:
    while True:
        try:
            a = pickle.load(f)
        except EOFError:
            break
        else:
            d.update(a)
```

In [ ]:
```
#save the number of articles that are referenced
total = 0
for url in url_list:
    total += len(d[url])
print(total)
```

```
148237
```

(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:

```
from scipy.sparse import csr_matrix
```

In [ ]:
```
from sklearn.feature_extraction.text import CountVectorizer
from scipy.sparse import csr_matrix
```

In [ ]:
```
documents = [' '.join(words) for words in d.values()]
```

In [ ]:
```
vectorizor = CountVectorizer()
sparse_matrix = vectorizor.fit_transform(documents)
feature_names = vectorizor.get_feature_names_out()
sparse_matrix.toarray()[0]
```

Out[ ]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [ ]:
```
import pandas as pd
import numpy as np
df = pd.DataFrame.sparse.from_spmatrix(sparse_matrix)
df = df.iloc[:, :24234]
```

In [ ]:
```
df_transform = df.div(df.sum())
```

In [ ]:
```
df_transform
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	...	24224	24225	24226	24227	24228	242
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
24229	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
24230	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
24231	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
24232	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
24233	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	

24234 rows × 24234 columns

(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 Sinhalese wikipedia.

In [ ]:
```
r_prev = np.ones(len(url_list))/len(url_list)
```

In [ ]:
```
sparse = csr_matrix(df_transform.values)
```

In [ ]:
```
beta = 0.85
c = (1-beta)* r_prev
r_new = r_prev
for i in range(40):
    r_new = beta * sparse.dot(r_prev) + c
    r_prev = r_new
```

In [ ]:
```
df_final = pd.DataFrame(r_new, columns=["Page Rank"], index = url_list)
```

In [ ]:
```
df_final.sort_values(by='Page Rank', ascending = False).head(10)
```

Out[ ]:

%E0%B6%B8%E0%B7%84%E0%B7%8F%E0%B6%AF%E0%B7%8A%E0%B7%80%E0%B7%93%E0%B6%B4%