CS F469 – Information Retrieval
Assignment-2A: Page ranking of web graph

Submission date & time: 9AM, 01-05-2022

General Instructions

1. This assignment is a coding/implementation project and is expected to be done in groups. Each group can contain at most **three** members. All members in the group should be registered for this course and try maintaining the same group for the assignments of this course.
2. This assignment is expected to be done in Python using standard libraries like NumPy, Matplotlib, NLTK and Pandas. You can use Jupyter Notebook and any other python in-built data structure or library. No other ML libraries like scikit/sklearn, TensorFlow, Torch etc., should be used.
3. All the **assignments would be run through a plagiarism detector,** and any form of plagiarism will not be tolerated and shall be brought to the notice of AUGSD/AGSRD. The final decision lies in the hand of the instructor, and only one submission per group would be allowed for one assignment.
4. All deliverable items (See below) should be put together in a single .zip file. Rename this file as FD<id-of-first-member> (preferably ID number of the student submitting) before submission.
5. Submit the zip file on CMS/Google Forms on or before the deadline, as mentioned above. The demos for this assignment will be held later, which shall be conveyed to you by the IC. All group members are expected to be present during the demo.
6. In case of any queries, please fill out the form. The responses will be shared on this doc.
7. You can also reach out to *f20180185@hyderabad.bits-pilani.ac.in* in case of any queries regarding this assignment.

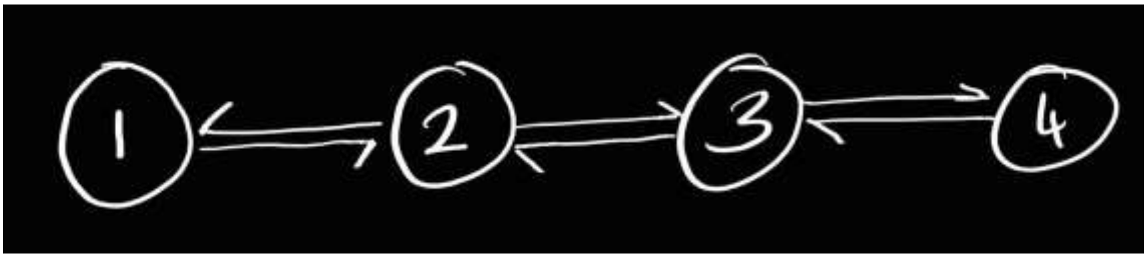
Problem Statement

1. This assignment is aimed at implementing the **PageRank algorithm** from scratch.
2. The PageRank algorithm should be implemented with and without random teleportations using the following two methods –

   A. Finding the **principal left eigenvector** of the probability transition matrix directly i.e., by making use of numerical linear algebra packages

   B. Finding the **principal left eigenvector** of the probability transition matrix **Power Iteration** method.

   You should implement Random Teleportations with the following hyper parameter: Teleportation to a random page with probability of 0.1
3. Feel free to use any input-output format and functional/class-based approach, as long as the README is proper (*point 3* of Deliverables).
4. The directed graph should be represented as per the below example and any other representation will not be considered for evaluation.



For the above graph the representation should be as follows:

```
//number of nodes: 4

n : 4

//number of connections: 6

e : 6

//edges are as follows

1, 2

2, 1

2, 3

3, 2

3, 4

4, 3
```

5. Try to vectorise your code as much as possible to make your computations faster and more efficient. Do not hard code any parts of the implementation unless it is indispensable.

Deliverables

1. **Code** – The code should be well **commented,** and all methods/classes for each implementation should be submitted.
2. **Documentation** – All the code's classes, functions, and modules must be **documented**, 'pdoc' is one such library to streamline this. The run-time of **PageRank** with respect to the number of edges/links should also be documented/plotted.
3. **README** – The README file should **describe** the **procedure** to run your code.

CS F469 – Information Retrieval
Assignment-2B: HITS

Submission date & time: 9AM, 01-05-2022

General Instructions

1. This assignment is a coding/implementation project and is expected to be done in groups. Each group can contain at most **three** members. All members in the group should be registered for this course and try maintaining the same group for the assignments of this course.
2. This assignment is expected to be done in Python using standard libraries like NumPy, Matplotlib, NLTK, Pandas and **NetworkX**. You can use Jupyter Notebook and any other python in-built data structure or library. No other ML libraries like scikit/sklearn, TensorFlow, Torch etc., should be used.
3. All the **assignments would be run through a plagiarism detector,** and any form of plagiarism will not be tolerated and shall be brought to the notice of AUGSD/AGSRD. The final decision lies in the hand of the instructor, and only one submission per group would be allowed for one assignment.
4. All deliverable items (See below) should be put together in a single .zip file. Rename this file as FD<id-of-first-member> (preferably ID number of the student submitting) before submission.
5. Submit the zip file on CMS/Google Forms on or before the deadline, as mentioned above. The demos for this assignment will be held later, which shall be conveyed to you by the IC. All group members are expected to be present during the demo.
6. In case of any queries, please fill out the form. The responses will be shared on this doc.
7. Find the Web Graph Dataset *here*.
8. You can also reach out to *f20180185@hyderabad.bits-pilani.ac.in* in case of any queries regarding this assignment.

Problem Statement

1. This assignment is aimed at implementing the **HITS algorithm** from scratch.
2. The HITS implementation expects the near-steady state values of the **Hub & Authority scores** of the nodes in the given web graph.
3. The following functionality is required to be implemented:

   a Get the **Hub & Authority Scores** of each node in the ***base set*** for a particular **query**.
4. You can find reference to *base set* & *root set* for a query in the textbook section 21.3.1 (pg. 477)
5. Feel free to use any input-output format and functional/class-based approach. You are expected to implement the HITS algorithm from scratch while using the libraries helper functions (not networkx.hits). Please ensure proper description of the README (*point 3* of Deliverables).
6. The dataset is a **Directed NetworkX Graph** with web content (string) saved in a GraphPickle (*web_graph.gpickle, point 7* of General Instructions) file which needs to be loaded in the python runtime by following code snippet given below. The graph has 100 webpages/nodes with 256 directional connections. You will need to **install** this library (if not already installed) by *pip install networkx* in the command prompt.

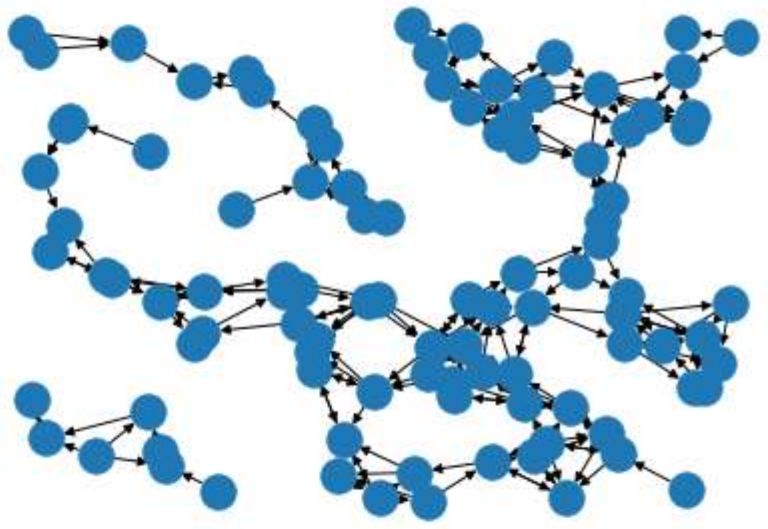
```
import networkx as nx #pip install networkx
web_graph = nx.read_gpickle("web_graph.gpickle")
web_graph
>>> <networkx.classes.digraph.DiGraph at 0x7f2e060629d0>
```

7. The *page_content* of each node/webpage can be accessed by:

```
#get the 50th page content
node_index = 50
G.nodes[node_index]['page_content']
>>> 'Sports: Giddy Phelps Touches Gold for First Time Michael Phelps won the gold
medal in the 400 individual medley and set a world record in a time of 4 minutes
8.26 seconds.'
```

8. To visualize (optional) you can also run the following which gives the graph figure with directed edges connecting nodes.

```
pos = {i: web_graph.nodes[i]['pos'] for i in range(len(web_graph.nodes))}
nx.draw(web_graph, pos)
```



A better **representation/visualization** of the same graph can be found on *this HTML file*. This can be viewed by downloading the HTML and opening with a browser (with page content on each node).

9. Try to vectorise your code as much as possible to make your computations faster and more efficient. Do not hard code any parts of the implementation unless it is indispensable.

Deliverables

1. **Code** – The code should be well **commented,** and all methods/classes for each implementation should be submitted.
2. **Documentation** – All the code's classes, functions, and modules must be **documented**, 'pdoc' is one such library to streamline this. The run-time of **PageRank** and **HITS** with respect to the number of edges/links should also be documented/plotted.
3. **README** – The README file should **describe** the **procedure** to run your code.