Asmt 8: PageRank

Turn in through Gradescope by 11:59 PM on Wednesday, April 26:
100 points

Overview

In this assignment, you will implement the Power Iteration Method with "teleportation" for finding PageRank importance scores of 10,000 webpages.

Report implementation only where asked.

Note: Homework assignments are intended to help you learn the course material, and successfully solve mid-term and final exams that will be done on paper in person.

As usual, it is recommended that you use LaTeX for this assignment. If you do not, you may lose points if your assignment is difficult to read or hard to follow. The latex source for this homework is available on Canvas. I recommend that you drop the two latex files (.tex and .sty) in a new Overleaf project, and compile/edit the .tex file there.

1 PageRank (100 points)

You are going to use the data in the file `web-Google_10k.txt` provided on Canvas and the following code (also available on Canvas) to load the data. **Read the code comments carefully!**

Nodes represent web pages and directed edges represent hyperlinks between them. There are 10,000 nodes and 78,323 edges in this graph. This graph represents a portion of the data that was released in 2002 by Google as a part of the Google Programming Contest. You can study the entire data with 875,713 nodes and 5,105,039 edges in the bonus problem.

```
filename = 'web-Google_10k.txt'
with open(filename,'r') as input_file:
    # The first 4 lines are metadata about the graph that you do not need
    # After the metadata, the next lines are edges given in the format: `node1\tnode2\
    n` where node1 points to node2
    lines = [item.replace('\n','').split('\t') for item in input_file]
    edges = [[int(item[0]),int(item[1])] for item in lines[4:]]

    nodes_with_duplicates = [node for pair in edges for node in pair]
    nodes = sorted(set(nodes_with_duplicates))

    # There are 10K unique nodes, but the nodes are not numbered from 0 to 10K!!!
    # E.g. there is a node with the ID 916155
    # So you might find these dictionaries useful in the rest of the assignment
    node_index = {node: index for index, node in enumerate(nodes)}
    index_node = {index: node for node, index in node_index.items()}
```

A (10 points): For each node, calculate the number of edges leading away from it ("out links"). Report your implementation, but do *not* report the values.

B (10 points): Find the dead-end nodes. Dead-end nodes were defined in the class. Report your implementation, the number of dead-end nodes in the graph, but do *not* report the IDs of nodes that are dead-end nodes.

C (70 points): Construct the transition matrix M and set the column of each dead-end node to make a uniform distribution over *all* the nodes. Implement the power iteration method with "teleportation" for calculating PageRank scores of nodes in the graph. Use $\beta = 0.85$ and $\varepsilon = 0.0001$. Report your implementation and the node (not its index, the node's original ID in the file `web-Google_10k.txt`) with the highest PageRank score.

D (10 points): How many nodes point to the node with the highest PageRank? Does that seem reasonable to you, and if so, why?

Bonus: Efficient Computation of PageRank (5 points)

You are going to use the full version of the graph in the first problem that is available here: `https://snap.stanford.edu/data/web-Google.html`. Download `web-Google.txt.gz` at the bottom of the page. This graph has 875,713 nodes and 5,105,039 edges. Explore some of the ways of speeding-up PageRank that are described in the "Mining of Massive Datasets" book in Section 5.2 `http://infolab.stanford.edu/~ullman/mmds/ch5.pdf`. Report your implementation and findings.