HW5 Step 2- Penn PageRank

Summary

Our idea was to do our own version of PageRank on a much smaller scale. The project involves starting from a “base” website, and searching for any links contained on the webpage. We then recorded those links on the base webpage as an edge from the base to that link, and repeated this process from all of those links. Once we had the full graph of webpages and outgoing edges, we performed the PageRank algorithm on the graph to find the importance of each webpage in the network.

Categories

The categories we used were:

- Information Retrieval
    - Our project requires the scraping of websites to find the outgoing links of each website.
- Information Networks
    - Our project deals with the structure of the World Wide Web, and how websites connect and interact with each other.
- Graph and Graph Algorithms
    - Our project requires the use of PageRank, which is a graph algorithm which uses a directed graph to determine the importance of a certain node.

Work Breakdown

We split up the work pretty evenly in two pretty manageable parts:

- Neil- The scraping of the websites. Finding the outgoing links for each website and recursing on those links. Should output an edge list of the resulting links.
- Joan- The actual PageRank algorithm. Converting the edge list into a usable graph representation, and running the PageRank on this new graph.

Results

We created 5 small internet graphs using 5 different starting URL’s: google.com, spotify.com, piazza.com, upenn.edu, and coronavirus.gov. We then ran the basic pageRank algorithm on each graph and recorded the ranks in decreasing order. For all the starting URL’s except for `https://google.com/` and `https://www.usa.gov/coronavirus` the ranks were distributed in a way that most nodes received a rank which was not lower than 10^{-2}. However, some of them had a rank in the range of 10^{-5} or 10^{-6}. That could have happened because those nodes were probably leaves (or nodes with a low degree) in the underlying graph and so the low number of neighbors

resulted in their small rank. For the `https://www.usa.gov/coronavirus` the ranks were more distributed, i.e. there were not jumps as large as the one described above (from 10^{-2} to 10^{-5}). That could be explained by the fact that the graph generated had probably nodes with a variety of degrees thus resulting in the ranks being more spread out. Finally, for the url `https://google.com/` the generated graph consisted only of 3 nodes. Additionally, the final ranks resulted in the node “blogger.com” having a rank of almost 1 and the remaining two a rank of almost 0. That would mean that in the underlying graph the “blogger.com” node was probably a sink.

User Manual

There are two separate parts of our project, the web crawler and the PageRank algorithm.

- The Web Crawler
    - To run this part of the project open a terminal in the root directory and run:
      ```
      $ node crawler.js
      ```
    - This will run the crawler on `https://piazza.com/` and create a text file called `edge_list_piazza.txt` containing an edgelist, representing a webpage and the links it contains on its page. A snippet of piazza’s edge list is to the right. There are actually 183 edges in the graph.
    - Feel free to change the website we crawl by changing the parameter to crawl on line 73, or the name of the file we write to on line 67.

```
piazza.com essentialaccessibility.com
piazza.com youtu.be
piazza.com calendly.com
piazza.com medium.com
piazza.com entrepreneur.com
piazza.com nytimes.com
piazza.com forbes.com
piazza.com npr.org
piazza.com fortune.com
piazza.com chronicle.com
piazza.com usnews.com
piazza.com twitter.com
piazza.com facebook.com
essentialaccessibility.com piazza.com
essentialaccessibility.com kentplace.org
essentialaccessibility.com thesource.ca
essentialaccessibility.com samsung.com
essentialaccessibility.com toysrus.ca
essentialaccessibility.com bestbuy.ca
essentialaccessibility.com facebook.com
essentialaccessibility.com twitter.com
youtu.be google.com
medium.com piazza.com
medium.com twitter.com
entrepreneur.com entrepreneurmedia.com
entrepreneur.com entrepreneurreprints.com
entrepreneur.com twitter.com
entrepreneur.com liveplan.com
entrepreneur.com greenentrepreneur.com
entrepreneur.com hearstmags.com
entrepreneur.com entm.ag
entrepreneur.com facebook.com
entrepreneur.com linkedin.com
entrepreneur.com instagram.com
entrepreneur.com youtube.com
nytimes.com facebook.com
nytimes.com twitter.com
nytimes.com piazza.com
nytimes.com nytco.com
nytimes.com nytmediakit.com
nytimes.com tbrandstudio.com
npr.org verb8tm.com
npr.org facebook.com
npr.org twitter.com
npr.org instagram.com
```

- PageRank
    - To run the page rank first load the Reader.java and Graph.java files.
    - Then in the Reader.java file the main method will contain 5 blocks of code of the form:
      ```
      Graph g1 = readFile("edge_list.txt");
      Map<Integer, String> originalInvIndex = new
      HashMap<Integer, String>();
      originalInvIndex.putAll(invIndex);
      ```
      Each of the 5 blocks of code mentioned above has a matching block of code of the form:
      ```
      printRanks(g1, originalInvIndex);
      System.out.println();
      ```

These pairs of blocks will first create the graph (1st block) and then run pageRank and print the ranks in descending order (2nd block).

- Then simply (un)comment the pairs of blocks corresponding to the desired graph. Note that the file named “edge_lists.txt” corresponds to the graph with the starting url being `https://www.upenn.edu/`.

- Finally run the main method which will print the final ranks for the chosen graph(s). If you wish to try other url’s feel free to add a matching pair of blocks of code like the ones mentioned above with the desired filename.