PAGE RANK ALGORITHM

Dinesh Kannan

kannan.d@husky.neu.edu

SUMMARY:

The goal of this module is to implement the first version of PageRank algorithm to assign weights to web pages based on their link structure. This weight is the probability that a random surfer would visit a page at a given moment, which is the PageRank of the page. The collection of webpages used for this is the WT2G collection. The program takes in the link structure of the collection as an inlink graph, where there is a page and a list of links pointing to it. This is implemented in Python and requires no specific external libraries to be installed.

MODULE SPECIFICS:

This module takes into consideration only the pages available in the WT2G collection and not the whole web itself. Hence there are pages without outlinks in the collection. In this condition, the rank of the pages is evenly distributed to all the other pages in the collection.

The **teleportation value d** is taken as **0.85**, which theoretically includes the probability that the random surfer does not push the "Surprise me button".

This module takes the inlink graph of WT2G collection as the argument. This is supplied into the program as the file "wt2g_inlinks.txt"

The perplexity value of the distribution is used as the test for convergence. The probabilities have converged if the difference in perplexity value is less than 1 for at least four consecutive iterations.

DESIGN:

IMPLEMENTATION:

The program has been split into the following modules, each serving a specific functionality.

- pageRank
- uniquePages
- processMatrix
- pageOutlinks
- inLinksCount
- rankResults

1. **pageRank :** This is the main module of the page rank functionality. It takes the help of other sub modules for performing small tasks. After these tasks, the module resumes with the main PageRank algorithm where it has two functions, pageRank() and pageNotConverged().
   The pageRank function starts with assigning equal probabilities to all the unique pages in the collection and iterates through list to determine the PageRank of all pages with the page rank formula until they converge. The page ranks of the sink nodes (pages without outlinks) are evenly distributed to all the pages in each iteration. Convergence is measured by checking if the difference in perplexity value of the distribution is less than 1 for atleast four consecutive iterations. pageRankNotConverged() function checks for the convergence of the probabilities, returns false when they have converged.

2. **uniquePages :** This module takes in the inlink graph and returns a list of unique pages in the collection.

3. **processMatrix :** This module takes in the inlink graph and returns a dictionary where each page is mapped to the set of inlinks to that page. Duplicate inlinks are removed when the dictionary is built.

4. **pageOutlinks :** This module takes in the inlinks dictionary to find the number of outlinks from each page in the collection. This is also returned as a dictionary, where the number of outlinks is mapped against a page.

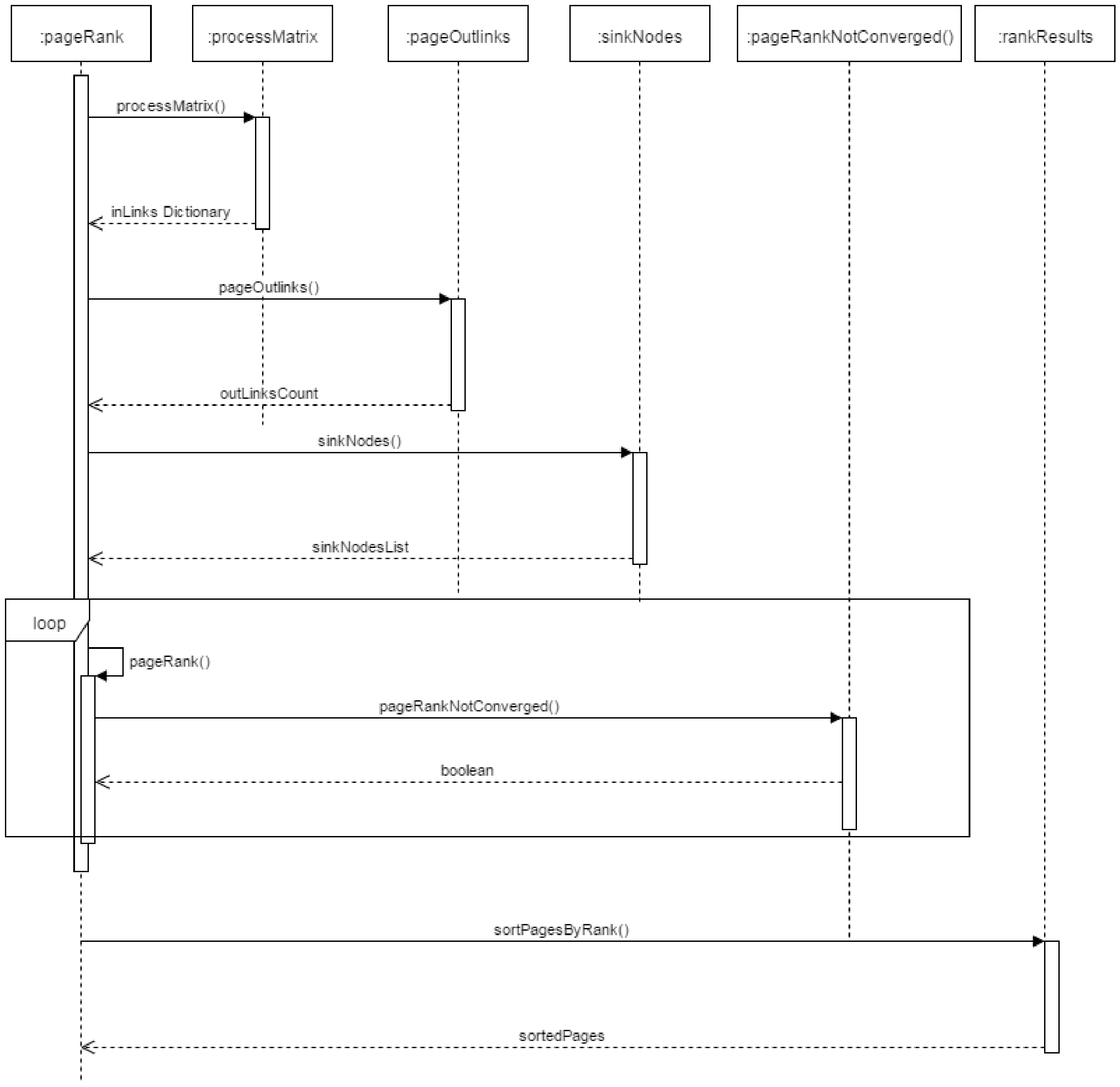
5. **inLinksCount :** This module counts the number of unique inlinks for each page. This step is not necessary for the PageRank algorithm, but is very useful for comparing results.

6. **rankResults :** This module is run after obtaining final page rank values for all pages, i.e after the page ranks have converged. This module takes in the final page ranks and sorts the pages according to their page rank values.

CONCEPTUAL MODELLING/ARCHITECTURE:

As mentioned above, this implementation is modular where each module serves a specific task. To better understand the conceptual modelling or architecture a UML Sequence Diagram of the implementation is depicted below. The flow and the discrete steps of the program is represented in this UML Sequence Diagram.

UML SEQUENCE DIAGRAM:



RESPONSES AND ANALYSIS:

To the question given in the assignment,

Question 1: Speculate why these documents have high PageRank values, i.e., why is it that these particular pages are linked to by (possibly) many other pages with (possibly) high PageRank values.

Response :

The top ten pages with respect to page rank and inlinks have high page rank values because, they **are most popular and important pages (The Economist, Security assurance requirements, Streetlink Financial reports etc..)** . Since they are more popular and public pages, they are assumed to contain authentic and useful information. Hence they are referenced by many other pages inorder to provide more information on a topic. These popular pages are inturn referenced by other popular pages as well (**eg. Toronto Dominion Bank**). Hence they have a higher page rank interms of more inlinks and also inlinks from pages with high page ranks.

Question 2: Are all of these documents ones that users would likely want to see in response to an appropriate query?

Response :

It may not be always that users would likely want to see these, because **there is no relation between page ranks of pages and the query**. For eg. If the user types in "Economics", not always would he want the home page of " The Economist" to come up. **Page ranks depend only on how the web pages are linked to each other and the importance of those linked web pages (link structure)**.

Question 3 : Give some examples of ones that are and ones that are not.

Response :

Consider the user query , " **Economics news**". Now the user would most probably expect articles from " The Economist", which has a high page rank. On the other hand, if the user queries for "**Copyrights**", he would most probably expect a general definition from Wikipedia and not the page "The Economist: Copyright Notice" which is the second highly ranked page in the collection. **This shows that relating page rank with the user query is not a good idea**.

Question 4 : For those that are not "interesting" documents, why might they have high PageRank values?

Response :

For those that are not interesting documents, like "The Economist: Copyright Notice", "Security Assurance requirements", the page rank must be because of their inlinks from higher page rank pages. That is, all pages of "The Economist" will contain a link about its copyrights at the bottom. Similarly, important pages will refer to security assurance requirements. Hence those pages, though uninteresting has high page rank values.

Question 5 : How do the pages with high PageRank compare to the pages with many in-links?

Response :

The top 2 pages are the same in both the categories in the results obtained. i.e. The Economist and The Economist : Copyright Notice . These two pages are the top with respect to both page ranks and the number of inlinks. **This does not necessarily mean that more inlinks correspond to more page rank**. A page with lesser inlinks can still have a greater page rank if its inlinks are from pages with higher page rank. For example, the Security Assurance Requirements page is ranked third in page ranks but is ranked 215th in inlinks count. But its fewer inlinks may be from pages with high page rank and hence its high page rank. This provides substantial ground that high pagerank doesn’t mean higher inlink count.

RESULTS:

The resulting page ranks for the given small input matrix is available in ‘**smallMatrix_pageRanks.txt**’.

For the WT2G Collection look at the following files,

- top50_PageRanks.txt : Contains the top 50 pages sorted according to PageRanks
- top50_inlinks.txt : Contains the top 50 pages sorted according to inlinks count
- otherResults.txt : Contains other calculations like perplexities at each round and the values of the proportion of pages asked in the problem set.

The analysis part of the implementation and the speculation part is done in the previous section ‘RESPONSE AND ANALYSIS’

CALCULATION:

PROPORTION OF PAGES WITH NO IN-LINKS**:**	**0.07806932120493333**
PROPORTION OF SINKS**:**	**0.3600165387272797**
PROPORTION OF PAGERANKS LESS THAN UNIFORM PROBABILITIES**:**	**0.7851434353765553**