CSI 2110

Assignment4

Group member

Waffo Tony Adams (300310088)

TA

Tianhao Tao

APRIL 12, 2023

OTTAWA UNIVERSITY

a. Description of classes and methods

We implemented 2 main classes in the code provided.

- GraphReader: This class allows us to get all data from the specified file and create an adjacency list with all nodes and their outgoing nodes (represented by edges using the readGraph method). Then it prints the 10 most influential nodes, which means the nodes with the highest PR's value.
- Graph: This class helps us constructing the proper graph using the nodes and edges from the adjacency list. It also provides us some methods such as getGraphNodes, getGraphEdges and the calPageRank method.

In these classes mentioned above, 3 methods are already given:

- getGraphNodes: gives all the nodes of the current graph.
- getGraphEdges: give all the edges of the graph.
- readGraph: read data from a file and create an adjacency list to represent the graph. Using the calPageRank method, it also prints the 10 most influential nodes (or pages) in the graph (or the internet).

On top of that, we must implement the **calPageRank** method. This method calculates the pageRank of all the nodes in the graph until the values of every Page stabilize. In other words, it calculates the final PR of the nodes in the Graph.

b. Description of the calPageRank algorithm

As mentioned earlier, the aim of this method is to calculate the PR (or value) of every node in the graph to later determine the most influential ones. The resolution process will be to assign each node an initial PR's value of 1 and calculate their PR until these values stabilize or don't change. To do a such think, we first create two HashMap, one holding the current PR of all nodes, and the other one called prevpageRank holding the previous PR of these nodes. Then for each node of the graph, we go through all the edges of the adjacency list to verify if it is referred by any other node. For each "referral" found, we divide its PR by the number of all its outgoing nodes (values.size) and add that partial result to the variable "partialResult" which is initialize at 0. After doing that for each referral node, we will multiply the partial result by the damping factor and add the value 1-d (where d is damping factor). By repeating these steps for each node, we will have the PR of each of them as expressed in the formula PR(A) = (1-d) + d (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn)).

Also, every node has his PR calculated several times, reason why there is a while loop. This loop will end up when all PR's value doesn't change anymore. We check the end of iterations by calculating the difference between the previous and current PR's value of each node to make sure the convergence's point is reached.
If the HashMap is stabilize, we just return it so that the result can be printed.