PageRank Documentation

Author: Ke Wang
GTID: 903058889
Email: kwang337@gatech.edu

PageRank Outline

PageRank algorithm is based on the following equation:

$$PR(A) = \frac{1-d}{N} + d\left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \cdots\right).$$

'd' represents alpha, and it equals to 0.85 in the implementation. Initial value is set to 1/N. And assume that each sink node can randomly jump to any node including itself.

My PageRank is implemented in Java. It requires Java SE 8 environment, because it uses some features not introduced before that version.

My implementation includes a default package and 'pagerank' package.

In the default package is 'Main.java' file, which is the main entry of the program. Its only function is to call PageRank to start.

In the 'pagerank' package are 'Edge.java', 'Node.java', 'PageRank.java' files. 'Edge.java' implements a directed edge, which represents a hyperlink of the web page. 'Node.java' implements a node, which represents a web page. 'PageRank.java' has a method *start()* which is the main method in the whole program.

The following is the details of the classes in the 'pagerank' package.

Class Documentation

Class Edge

Each hyperlink is represented by a directed edge in the graph. It has two end nodes references. The class Edge has a method *getAnotherNode(Node n)* which gets the other end of the edge other than n, and if n is not the incident node, null is returned.

Class Node

Each web page is represented by a node in the graph. The node contains an id, a PageRank value, a temporary value, incoming edges and outgoing edges. The PageRank value is the current value of the node (web page). The temporary value is used when calculating the PageRank value. Because we need to simultaneously update the PageRank value after one iteration, the temporary value field is needed to save the value before update. The class Node also contains a few getters and setters methods.

Class PageRank

Class PageRank is the main class of the program. It contains two final fields PRECISION and ALPHA. PRECISION is set to 0.000001 by default. If the max difference between two iterations of the algorithm is less then PRECISION, then we consider it has converged. ALPHA is 'd' used in the up-mentioned formula. It is set to 0.85 by default. The class contains only one method *start()*, which starts the PageRank algorithm. It first reads input from *System.in* while initializing a node list and edges. Then it updates the value iteratively using the formulation above while the max difference between the iterations is larger than PRECISION. As last, output the results to *System.out*. The details of *start()* is described below.

The method uses *java.util.Scanner* to scan the input. When it reads in the number of nodes N, it creates a node list and initializes it with N nodes. Then it reads in the edges and updates the N nodes correspondingly. Next is the iteration step (a while loop) of the algorithm. An *OptionalDouble* is used to maintain max difference between the two iterations. If the difference value is smaller than PRECISION, then the loop ends. The loop contains three parts. The first part reset the *tempValue* field in each *Node* class. This field is used when calculating the new PageRank value for each node before updating the value simultaneously. The second part calculates the value each node gives out. If one node has no outgoing edges, then it is a sink node. It gives equal value to all nodes including itself. If one node has outgoing edges, then it gives its value to the nodes on the other end of each edge equally. The last part calculates the final value of current iteration. Each node collects the value given to it by other nodes and uses ALPHA to calculates the updated value. The difference between the two iteration is maintained in an *OptionalDouble* variable. At the last of the method, all PageRank values are printed out to *System.out.*