PageRank

Project Due: 8:00 am EST Monday February 25, 2018

In this assignment you will implement the PageRank algorithm used for ranking webpages. You will be provided with two small and one large web graph datasets to run your implementation of PageRank on, as well as to verify your results. You will then analyze the performance and stability of the algorithm as you vary its parameters.

Restrictions

- You must complete this assignment on your own, not in a group.
- You must use **python (2.7.x)** to implement your project. Template code is provided and must be used.
- You may use any built-in python functions, functions provided by libraries already imported in the template code, and any plotting libraries, such as matplotlib. You may not use any other libraries or code from any other source.

  A Note about Numpy : you will notice that Numpy has been included in the template code. You may use Numpy if you wish, **but you do not have to.** You can easily meet the requirements of the project using the provided data structures.
- All code modifications/additions that you require must be put in the "PageRank.py" file, as this is the only python file accepted for submission.
- You need to turn in a report (see below) to Gradescope, and a single source code file ("PageRank.py") via Canvas. Please do not submit an archive or .zip file, but rather individual files.

Template Code

You have been given template code in the form of two .py files to assist you in completing this assignment. The first file, "prProjectUtils.py" provides utilities for reading, writing and testing your code, while the second file, "PageRank.py" provides the template code that you will modify to complete your project.

There are sections within the code in "PageRank.py" with the comments "#your code goes here" where you should put your code for each of the tasks. Do not modify the flow or structure of the template code, as this could impact the autograder. You will only turn in "PageRank.py" to Canvas, so if you need to write extra functions to run your code, put them in that file.

To execute your code, using the following command:

python pageRank.py -i <input_file> -a <alpha> -e <epsilon> (-s <0,1> optional -p <0,1> optional)

- input_file : ("testInput.txt") File name of the file holding the graph description, in adjacency list form. The entries in this file consist of a node id, followed by the node ids that it points to. The functionality to read this file and put it into a usable data structure (an adjacency list of out-edges) has been provided for you already, and you should not need to interact with the file otherwise.

- alpha : floating point value for alpha which should be positive and less than 1.0. For your report you will vary it between 0.75, 0.85 and 0.95

- -s : flag setting the sink handling of the project to be either self-loops (0) or type 3 (1). Default value is to not use self-loops but rather treat sinks as specified in video #26 method #3.

- epsilon : value to use for convergence checking of L_{inf} norm of PageRank vector during solver iterations. The default value of 1e-10 is sufficient, and this value you should not need to modify.

- plot : flag setting whether you wish to plot results or execute PageRank. Plotting results successfully is reliant on already generating the PageRank results for the 3 alpha values.

Tasks

You are tasked with implementing the PageRank algorithm, supporting two different strategies for handling sink nodes (giving every node a self-loop, and a method we will call "Type 3", based on the explanation in video #26. Sink Handling in the instructional videos). Your code must run correctly on the provided sample data sets, verified against the correct solutions which have also been provided. **Your code must execute on the "Large.txt" file within 8 minutes, using an alpha value of .85 and Type 3 sink handling (this is what it will be set to for grading)**. This time constraint will dictate certain design decisions you need to make, (for instance, you cannot have a probability matrix, since it will be too large).

The code you have been provided contains the framework of a PageRank class, along with functionality already implemented to perform numerous tasks such as accessing the input file specified by the command line and loading the data into an adjacency list, keyed by node id, where each value is a list of all nodes that the key node points ("out" list). The code to save and test the results of your algorithm is also provided and need not be modified, as is the code to provide the appropriate plots, which you should use.

You have been provided 2 small datasets, "testCaseSmall.txt" and "testCase3k.txt", as well as a large dataset, aptly named "large.txt". For each of these datasets, validation sets have also been provided for various settings of alpha and for both sink-handling strategies. Whenever you generate a PageRank vector, the results will be automatically validated against the appropriate validation datasets if they exist in your working directory.

The project has been parsed into three separate tasks to aid you in implementing your algorithm, which you should complete in order before you can conduct the experiments necessary for your report. These tasks are specified by three tags of "your code goes here <#>" in the "PageRank.py" template code file.

1. Initialize all essential data structures

You must put the code for this task in the "initAllStructs" method already specified in the template code. For this task you will be setting up the essential data structures you will need to solve PageRank that remain unchanging throughout the execution of the algorithm. For performance concerns these unchanging structures should only be built once, and this is what we will do in this task.

Note that in the PageRank class constructor you have already been provided with code that gives you a list of all the nodes in the map ("self.nodeIDs"), as well as a dictionary whose keys are all non-sink nodes and whose values are all the nodes the keys point to ("self.adjList"), which we'll call an "out list". For this task, you must do the following:

- Add all self-loops to self.adjList if self-loop sink handling is being used (do not do this if Type 3 sink handling is being used).
- Build a data structure that maps every node in the map with the nodes that point to it – this is the "in list".
- Build a data structure that records every node's out degree.
- Build a list of all sinks in the map, if Type 3 sink handling is being used (do not build if self-loops are being used).
- Initialize the PageRank vector structure properly (to a uniform distribution)

A note about the "in list" **- you cannot use an adjacency matrix** for this functionality as it will take up too much room on the large example. You must use some other structure to perform this functionality.

2. Solve Iteration of PageRank Algorithm

Task 2 is to solve an iteration of the PageRank algorithm. You must complete the code in "solveRankIter" method, which performs a single iteration of the PageRank algorithm, remembering to appropriately manage sinks. Your vector result after each iteration needs to be of unit magnitude, since it is a probability vector – if it isn't then there's something wrong with your calculation. You must also conditionally handle both sink-handling strategies correctly in this function. For "type 3" sink handling, a naïve approach might be to add every sink to every node's "in list" but this will result in a computational explosion and must not be done. Instead, there is a more efficient mechanism you can use that adds much less overhead but appropriately handles the sinks' contributions to all nodes' rank.

3. Solve PageRank Algorithm

Lastly, you must write the code to repeat the iterations of the algorithm until the convergence criteria have been met. This code must be put in the "solveRankToEps" method. We are wanting the L-infinite norm of the difference of two successive iterations of the rank vector to be less than **eps** ($\boldsymbol{\varepsilon}$) to determine convergence. This means that, for all elements x of a vector $\mathbf{V}$ at time T, $|V^{(T)}(x) - V^{(T-1)}(x)| < \boldsymbol{\varepsilon}$. In other words, we stop when no component of the rank vector changes more than $\boldsymbol{\varepsilon}$ after an iteration of the solver.

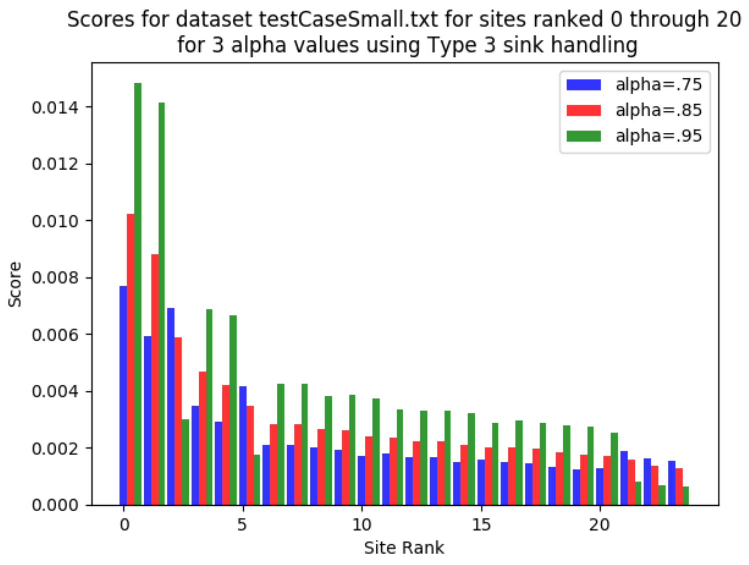
Your PageRank program will generate a single file that contains the rank vector values for all the nodes from the original input file, in sequential order (node ID 0's rank value is the first entry in the file, followed by node ID 1's rank value, etc.). This file will be validated against the appropriate validation file, which will have the same name but with the suffix "verifyRVec". If this file does not exist in the same directory as the pageRank python code, the verification will not take place.

Report

You need to turn in a report for this project, and a template file, "PageRankReportTemplate.odt" has been provided for you to use. Your report must follow the layout of the template, and you should only use the space provided for your analysis and plots. Your report should be submitted to Gradescope as a .pdf file.

The focus of your report will be measuring the impact of varying alpha (.75, .85, .95) on the calculation of the PageRank vector for both types of sink-handling strategies. You will accomplish this by showing how the top 20 sites in the large dataset are impacted by changing alpha and by using both sink-handling strategies. Note that since the top 20 sites may differ for different alphas, there may be up to 60 site results displayed – you will be showing the results for the union of the top 20 sites for the three values of alpha (.75, .85, .95) for each sink-handling strategy.

You will generate one plot for self-loop sink handling, showing alpha == .75, .85 and .95 on the same plot, and you will generate another plot showing the results for Type 3 sink handling with those same alpha values. The functionality to build the report graphs is already provided to you, you just need to make sure you have valid PageRank vectors built for the appropriate alpha values and sink-handling strategies. A plot run on one of the small datasets can be seen below – your results will have this general format.



For your report, you should put both plots of the results from the "Large.txt" dataset on the first page in the spaces provided. You should answer the following questions in the discussion section on the second page. Do not use more pages than those provided and keep the answers within the margins specified in the template document.

Questions to consider for your report write-up: What are the results you see in the plots? What do the values represent? What does increasing alpha toward 1.0 mean? Why do the results change with changing alpha? Why does changing alpha affect the runtime? How does the sink handling strategy impact the results and the runtime?

Piazza

All Q/A, discussions and announcements regarding the project will be monitored via Piazza, via the "project" tag. You will be expected to be aware of any official announcements regarding the project made on piazza. In an effort to assist you with this, relevant posts will be aggregated and pinned.

Since this class is so large, we ask that you assist us in keeping Piazza accessible and helpful by doing the following:

- Always add the "**project**" tag to any posts relating to the project.

- Search for existing posts that are topically similar to your question, and within these posts ask follow-up questions, rather than starting new posts with the same question, if possible.
- Post publicly – do not send the instructors private messages regarding the project unless - explicitly asked for.
- Do not publicly post code or your PageRank vector, etc.
- Do not discuss the implementation details or your results publicly on Piazza; if you're unsure about your question ask publicly if you can ask questions regarding specific details of your implementation.