

# MSAN 502 - Homework 4

*Andre Guimaraes Duarte*

*August 10, 2016*

## Google PageRank

### Wiki-Vote

The python file corresponding to this part of the homework is `wiki.py`.

In this exercise, we wish to analyze a connected network graph using the pagerank algorithm. For this purpose, I chose to use the *Wikipedia vote network* found on <http://snap.stanford.edu/data/wiki-Vote.html>. This dataset contains all the Wikipedia voting data from the day Wikipedia was launched until January 2008. In this network, nodes represent unique Wikipedia users. A directed edge from node  $i$  to node  $j$  represents that user  $i$  voted on user  $j$ . In total, there are 7115 nodes representing users, and 103689 edges representing votes. For this dataset, pagerank will allow us to see which Wikipedia user is the most important with regards to voting. The dataset is found in the text file `Wiki-Vote.txt`.

Note: Since the 7115 nodes are not perfectly labeled  $1 - 7115$ , and it would take a lot of work to uniquely replace the values with a perfect range, I decided to keep the original labels, which go from 1 to 8297. Consequently, my matrices are  $8297 \times 8297$  instead of  $7115 \times 7115$  (but this is ok).

I used a custom python code in order to parse the data set and construct the adjacency matrix  $A$ . My code goes through the edges in the text file and sets a value of 1 for the corresponding entry in  $A$ . Then, I make  $A$  column-stochastic in a few steps:

- first, I transpose the matrix  $A$ , because it is easier to work on rows;
- then, I divide each row by the sum of its entries, so  $A^T$  is row-stochastic;
- if the sum of a row is zero, I set the diagonal entry to 1;
- finally, I transpose the matrix in order to get  $A$  again.

At the end of this process,  $A$  is column-stochastic.

The following step consists of calculating the Google Matrix<sup>TM</sup>  $M = (1 - \alpha)A + \alpha B$ , as seen in class ( $\alpha = 0.15$ ).

Using `linalg` functions in `numpy`, it is easy (albeit time-consuming!) to compute the eigenvalues and eigenvectors of  $A$  and  $M$ . In the end, we obtain an eigenvalue  $\lambda = 1$ . Note: The exact value for  $\lambda$  computed by python is 0.9999999999999956, which we consider equal to 1 (there are floating-point approximation errors for example).

The pagerank vector, the eigenvector associated to this eigenvalue, is 8297-dimensional, so I won't print it here. However, I did compute the sorted indices of this vector in order to see the 10 first elements for several values of  $\alpha$ , as shown in the table below.

We can verify that the sum of all the values in the pagerank vector is equal to 1. Note: The exact value for this sum as calculated by python is 1.00000000000001377, which we consider equal to 1 for the same reasons as previously.

$\alpha = 0$		$\alpha = 0.15$		$\alpha = 0.5$		$\alpha = 0.75$		$\alpha = 1$	
ranked nodes	value	ranked nodes	value	ranked nodes	value	ranked nodes	value	ranked nodes	value
8211	1.0	2624	0.0078391	2469	0.003098	4036	0.001744	4036	0.000121
0	0.0	2469	0.0060250	2624	0.002926	2469	0.001504	2469	0.000121
1	0.0	7552	0.0051800	4036	0.002519	2624	0.001229	14	0.000121
2	0.0	1185	0.0048590	1185	0.002480	1185	0.001225	2236	0.000121
3	0.0	7619	0.0046120	8292	0.001968	14	0.001173	1185	0.000121
4	0.0	5411	0.0045810	7552	0.001842	2236	0.001053	2624	0.000121
5	0.0	7631	0.0045540	4874	0.001842	8292	0.000944	664	0.000121
6	0.0	4874	0.0044730	213	0.001839	213	0.000884	6773	0.000121
7	0.0	6831	0.0042210	7619	0.001809	4874	0.000845	8292	0.000121
8	0.0	2065	0.0040960	14	0.001796	7619	0.000796	2653	0.000121

For  $\alpha = 0$ , only matrix  $A$  is in play, and the eigenvector is  $(1, 0, 0 \dots)$ , where the highest-ranked node is 8211. This means that the steady-state is solely on this node. In the end, this user has sole influence on the votes.

With  $\alpha = 0.15$ , we can see that now all nodes (only the first 10 are displayed) have steady-state probabilities and are ranked. Node 2624 is the most influential in Wikipedia votes from its inception until January 2008.

When  $\alpha = 0.5$ , we can see that the ranking is different from the previous one. Node 2624 is now in second place, and 2469 is the user with most influence. We can also see that the dispersion in values is now smaller.

With  $\alpha = 0.75$ , the ranking has again changed, with node 4036 taking the first place. Matrix  $B$  has a bigger influence than  $A$ , and so the values are converging toward a uniform distribution.

When  $\alpha = 1$ , only matrix  $B$  is in play, and we can see that the steady-state is uniform. All nodes have the same weight equal to  $\frac{1}{8297} = 0.000121$ .

Since this dataset is very large and computing the eigenvalues and eigenvectors takes a long time, I also applied this program on a smaller dataset.

## Karate

In order to test varying values for  $\alpha$ , I also applied the pagerank algorithm to the **karate** dataset, found on <http://www-personal.umich.edu/~mejn/netdata/>. This dataset is a social network of friendships between 34 members of a karate club at a US university in the 1970s.

I had to format the data in away that the python code I wrote would work for this data, and the resulting file is **karate.csv**. I analyze it using the file **karate.py**.

The table below shows the nodes according to pagerank for several values of  $\alpha$ :

$\alpha = 0$		$\alpha = 0.15$		$\alpha = 0.5$		$\alpha = 0.75$		$\alpha = 1$	
ranked nodes	value	ranked nodes	value	ranked nodes	value	ranked nodes	value	ranked nodes	value
0	1.0	0	0.485245	0	0.229594	0	0.109926	0	0.029412
1	0.0	23	0.071663	23	0.052955	2	0.044278	2	0.029412
2	0.0	24	0.060736	1	0.048818	1	0.043845	1	0.029412
3	0.0	26	0.045022	24	0.046757	23	0.040740	23	0.029412
4	0.0	20	0.033269	2	0.046406	24	0.037721	24	0.029412
5	0.0	15	0.033269	26	0.038190	26	0.033660	26	0.029412
6	0.0	18	0.033269	15	0.031653	14	0.030523	5	0.029412
7	0.0	22	0.033269	18	0.031653	15	0.030523	14	0.029412
8	0.0	14	0.033269	20	0.031653	18	0.030523	15	0.029412
9	0.0	1	0.025904	14	0.031653	20	0.030523	18	0.029412
10	0.0	2	0.022129	22	0.031653	22	0.030523	20	0.029412
11	0.0	5	0.009318	5	0.023897	5	0.028722	22	0.029412
12	0.0	3	0.008209	3	0.022113	3	0.027594	3	0.029412
13	0.0	4	0.007443	4	0.020221	4	0.025965	4	0.029412
14	0.0	8	0.007111	8	0.019783	8	0.025753	8	0.029412
15	0.0	6	0.006287	6	0.018382	6	0.024816	6	0.029412
16	0.0	28	0.005693	28	0.017117	28	0.023814	28	0.029412
17	0.0	25	0.005472	25	0.016684	25	0.023490	25	0.029412
18	0.0	29	0.004990	29	0.015827	29	0.022892	29	0.029412
19	0.0	30	0.004990	30	0.015827	30	0.022892	30	0.029412
20	0.0	31	0.004990	31	0.015827	31	0.022892	31	0.029412
21	0.0	32	0.004632	32	0.015138	9	0.022383	9	0.029412
22	0.0	27	0.004632	9	0.015138	13	0.022383	13	0.029412
23	0.0	9	0.004632	13	0.015138	19	0.022383	19	0.029412
24	0.0	13	0.004632	19	0.015138	27	0.022383	27	0.029412
25	0.0	19	0.004632	27	0.015138	32	0.022383	32	0.029412
26	0.0	7	0.004412	7	0.014706	7	0.022059	7	0.029412
27	0.0	10	0.004412	10	0.014706	10	0.022059	10	0.029412
28	0.0	11	0.004412	11	0.014706	11	0.022059	11	0.029412
29	0.0	12	0.004412	12	0.014706	12	0.022059	12	0.029412
30	0.0	16	0.004412	16	0.014706	16	0.022059	16	0.029412
31	0.0	17	0.004412	17	0.014706	17	0.022059	17	0.029412
32	0.0	21	0.004412	21	0.014706	21	0.022059	21	0.029412
33	0.0	33	0.004412	33	0.014706	33	0.022059	33	0.029412

We can see that the ranking of the nodes varies with  $\alpha$ . When  $\alpha$  is zero, we are effectively computing the pagerank vector on the matrix  $A$ . We can see that the steady state is solely on the node 0, which has weight 1 whereas all other nodes have weight 0. By doing a linear combination with  $B$ , all nodes will have an importance in the final steady state.

With  $\alpha = 0.15$ , node 0 has a very high value compared to all the other nodes, but now all nodes have values in the pagerank vector. We can also see that the last 20 nodes have very similar low weights.

When  $\alpha = 0.5$ , the weight is better distributed along the nodes. Node 0 goes from a value of 0.485245 to 0.229594. In addition, the bottom-ranked nodes see their values increase. We can also see that the order of the nodes has changed, especially towards the tops of the table.

With  $\alpha = 0.75$ , the uniform matrix  $B$  has more weight than  $A$ , and we can see that the values for the nodes are converging toward  $\frac{1}{34} = 0.029412$ . Node 0 is still at the top of the ranking, but the difference between its value and the second-ranked node's value is significantly less than for  $\alpha = 0.15$ .

When  $\alpha = 1$ , we can see that the pagerank is run only on  $B$ , and all nodes have the same rank and the same value of  $\frac{1}{34} = 0.029412$ . The steady-state is uniform across the nodes.

## Conclusion

In this exercise, we have implemented Google's Matrix<sup>TM</sup> to a connected graph in order to compute the pagerank (eigen)vector associated to  $\lambda = 1$ . We have applied this algorithm to two distinct datasets, one large and one small, and have seen that it worked in both cases rather well. We were able to rank the nodes in each case by their importance in the network. For the small dataset, we were in addition able to vary the

value of  $\alpha$  and see its influence on the resulting pagerank vector. The more  $\alpha$  is big, the more the pagerank vector will be uniform. According to Google, the optimal value for  $\alpha$  is 0.15, so  $B$  introduces a little bit of stochasticity to the system, but not too much.