Laboratory 3

Ashwin Kumar Gururajan
Mohana Fathollahi

UE Information Retrieval and Recommender Systems (IRRS)

1 Introduction

In this lab we implemented a version of the Google page rank for the airport data set. Since it was intended to be initially used for ordering search results it needed to be efficient on large actual webgraphs and the relative ordering of terms was more important than their abosulte page ranks.

We implemented a version of the pageranks that accounts for spider traps and sink nodes/dead ends. The solution for spider traps was trivial using the damping factor while the sink nodes was a bit harder to tackle. In sink nodes we do not have outgoing links and when we apply summation over all page ranks in the graph, we will not reach to 1. It is because of existing sink nodes or dead ends.

Some corrections on how to tackle this problem have been suggested, one of them is to re-normalize page rank after every update. It has a problem of overestimating and underestimating page rank for nodes with high incoming links and low incoming links respectively. Therefore, this approach is obviously not an efficient way to solve this problem.

Another approach is adding outgoing links to the sink nodes. Although if we naively add these edges to our graph is blows up the number of edges in our originally sparse graph. A slightly better solution is to add these extra edges implicitly for sink nodes. Therefore, another term should be added to page rank formula for sink nodes and consider outgoing links implicitly for these nodes to all other nodes. The revised page rank formula will look as follows :

$$PR(x) = \frac{1-\lambda}{N} + \lambda \sum_{y->x} \frac{PR(y)}{out(y)} + \lambda \sum_{z->\emptyset} \frac{PR(z)}{N}$$

2 Implementation

The first initialization for page rank is assigning a random probability to all nodes we chose to initialize all nodes with an equal probability of $1/n$, which in our case equals, 0.00017. In every iteration, the value of the page rank is updated. We use Q to store the values of the page rank for each node in the current iteration and update it accordingly. If the difference in absolute page rank values between the current iteration and the previous iteration is less than the threshold value i.e if the page rank values stabilizes then we stop the iteration and display the results.

We have 2455 sink nodes whose page rank should be calculated based on the above formula and like other nodes their page rank should be updated in each iteration.

To reproduce the results run the pagerank.py file and to obtain the plots run the ipynb file.

3 Analysis

Different values for threshold and lambda (damping factor) has been considered and we analyzed them in the below figures ;

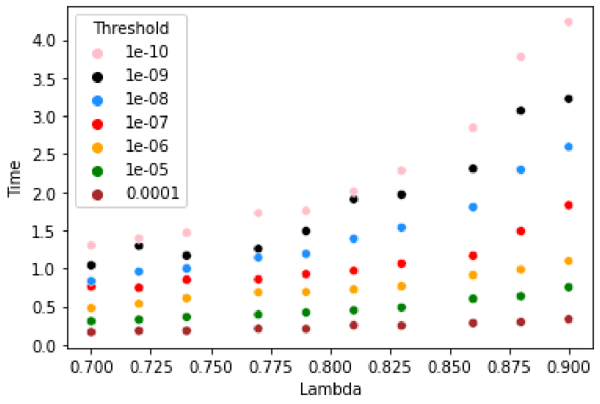

Figure 1. Time to converge for each lambda

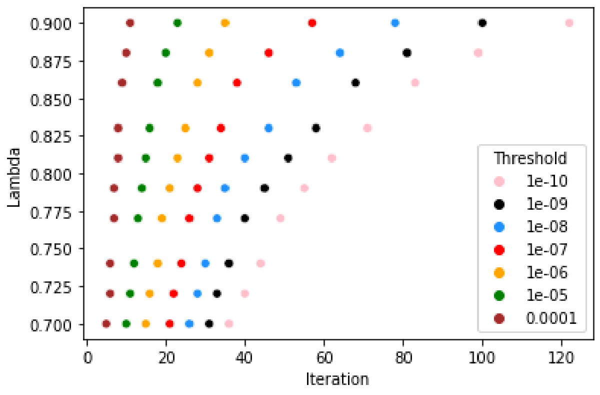

Figure 2. Number of iterations for each lambda

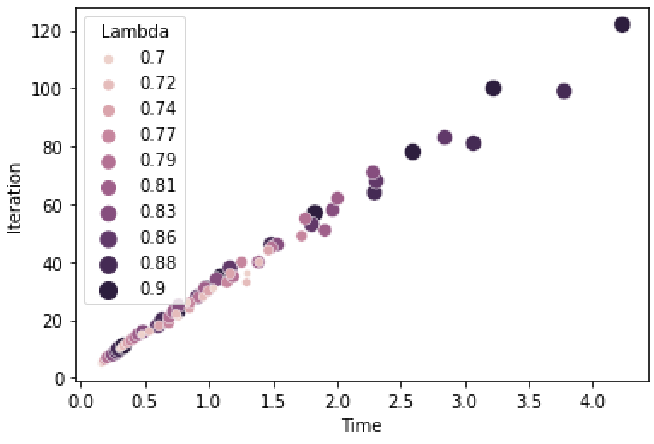

Figure 3. Time-iteration for each lambda

We considered 10 values for lambda that start from 0.7 to 0.9 and 7 values for threshold that start from 0.0001 to 1e -10. Based on the above figures, when we have small value for threshold and a small value for lambda we needed less time and iterations to converge. On the other hand, when we increased the threshold and lambda values we needed more time and iterations to converge. It makes sense that by considering a lower value for the threshold we are trying to consider a narrower interval for the difference between P and Q. Although lower threshold values converge faster we have to remember that we want to converge fast only after pagerank stabilizes. When we have an absurdly high value of thresholds such as 0.01 the algorithm converges in 1 iteration but it basically means that the algorithm didn't run.

Additionally, after analyzing the behavior of lambda we can say that, when lambda is very small the first term usually dominates the second term and the edges between nodes in the graph get less effective. As a result, the algorithm converges in less number of iterations. While if lambda increases, the second term dominates and the algorithm requires more iterations to converge since at each iteration the page rank of each node changes and it will change its neighbor's Pagerank too.

In the below plot, we are considering the number of income edges for each airport and compare it with the page rank of that airport. The general trend in this plot is that when we have more incoming edges we have a higher page rank. But we should notice that in some cases we have a higher page rank even with lower incoming edges, for example when one airport has 150 incoming flights its PageRank is higher than 0.003 but when we have 250 flights to another airport its page rank is less than 0.002. Therefore, the page rank of airports that points to the airport with 150 flights probably has higher values.

Another interesting observation is that the majority of the nodes are clustered in zero areas where a portion of them are sink nodes. This would confirm the importance of the third term in the page rank algorithm to support sink nodes.

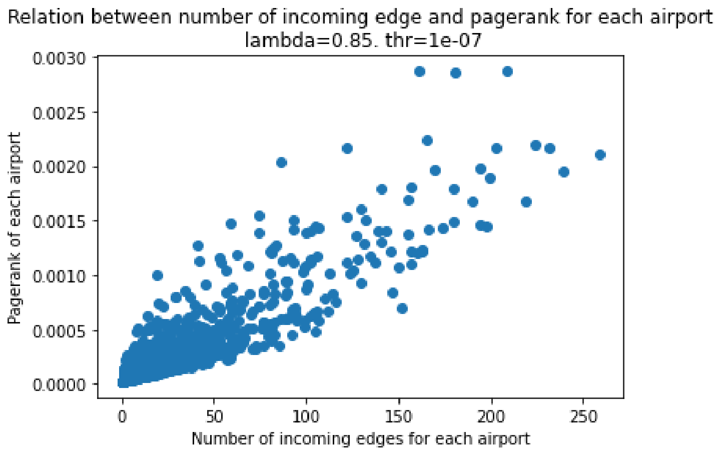

Figure 4. Incoming edges vs pagerank

In the next analysis, we considered different thresholds and calculate the percentage of mismatch in the ordering of the airports as it relates more to the ordering of the search results which the algorithm originally intended to solve. For example when the threshold changed to 10^{-6} number of mismatches i.e the difference between current ordering to previous ordering considerably reduced. On the other side when the threshold changed to 10^{-8} we did not have considerable changes in the number of mismatches. Therefore we can say the optimal value for the threshold in this dataset considering run time for a lambda value of 0.85 is 10^{-7}. This indicates that both the lambda parameter and the threshold value have a significant impact on the output ordering of page ranks and picking the right parameters is not a trivial task.

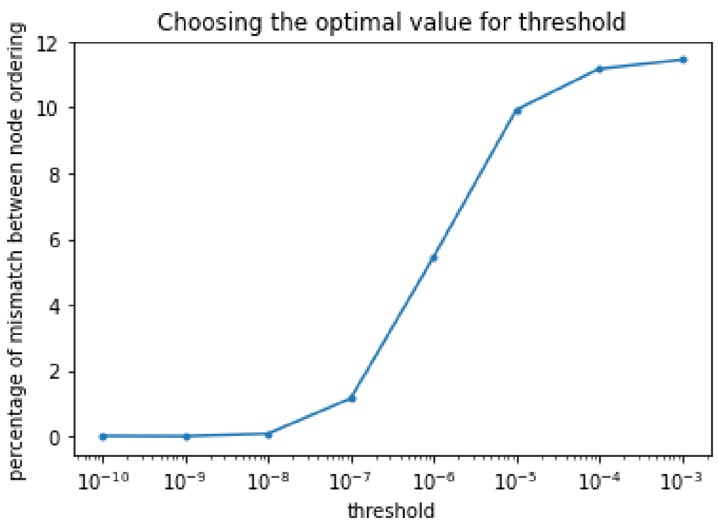

Figure 5. Choosing the optimal value for threshold

4 Conclusion

In the original paper, a web graph with over 3 million nodes converged in around 50 iterations. We observe similar behavior in our graph taking into account the threshold values.

That said the PageRank algorithm isn't the be-all and end-all. PageRank is very good at the overall relative ranking of nodes, although in cases where the page rank values are very close it struggles to achieve a fixed ordering. In practice, we've seen that the top 3 search results get more than 50% of the clicks and the ordering of the results plays a very huge role in the click rates. This is one of the reasons Google doesn't rely solely on the page rank algorithm anymore and has moved into more complicated methods for search ordering.