Information Retrieval and Analysis
OCTOBER 2020

Lab Session 6: Implementing Pagerank

PROJECT REPORT

Elías Abad Rocamora
Victor Novelle Moriano
Barcelona, UPC - FIB & FME

Implementation

When implementing the PageRank code we did not encounter any major problems rather than the usual bugs one usually finds.

In every iteration we use the list of outgoing airports to incrementally update the vector of weights. This is better than iterating the complete adjacency list and getting all the airports that have a connection with airport “X”, to definitely update “X”’s weight for that iteration, as we only need to load each row of the adjacency list once per iteration. But the weight of airport “X” won't be definitive until we finish iterating on the nested loop.
This leads to a more efficient use of disk accesses if the adjacency list doesn’t fit in our memory.

To avoid a non-termination of the algorithm, we have added as a stopping condition that the distance between the previous vector of weights and the current one is smaller than 10^{-6} or that the number of iterations is bigger than 1000. This is key when testing for d=1 as it sometimes doesn’t converge to a solution.

Airport page rank

After executing our code with the simple graph provided, and making sure the obtained results matched the ones given on the course slides, we proceeded to execute our algorithm using the airports’ graph. The obtained results did not surprise us at all. The airports with higher page ranks correspond to humongous ones, located in important cities, like *Chicago Ohare Intl* among others. The fact that the top results correspond with big airports is logic, because a bigger dimension generally implies a major number of arrival/departures. This fact is confirmed when the lowest ranked airports are observed. These airports are characterized for having runways full of potholes and high density of aircraft accident pictures when looking for the airport’s name in google, like *São Félix do Xingu*. Also, in the lowest ranked airports we found ones located in dangerous zones like the *Damascus international airport,* or remote zones, like the *Nauru airport*. These airports don’t receive too many arrivals, and we cannot guarantee that all of them are correctly registered after seeing the precariousness of the infrastructures.

Damping factor effect

In order to test the effect of the damping factor we will execute the code with values of d ranging from 0 to 1 in 0.05 steps. The results for execution time and number of iterations are written in this table.

d	0	.05	.1	.15	.2	.25	.3	.35	.4	.45	.5	.55	.6	.65	.7	.75	.8	.85	.9	.95	1
Time	0.044	0.241	0.310	0.354	0.447	0.492	0.632	0.648	0.769	0.799	0.798	1.090	1.233	1.447	1.758	2.232	2.826	3.817	6.160	12.331	37.185
Iterations	1	7	9	10	12	14	16	18	20	23	27	31	36	42	51	63	81	112	172	353	1000

As we can observe in this table, the number of iterations, and consequently, the execution time, grows with the parameter d, until the point that with d = 1, the execution is cutted at 1000 iterations, so it is likely that the execution doesn't terminate.

Bonus Track

By means of trying to compute the pagerank in some small undirected graphs, we've guessed that the final pagerank of a node i with out(i) connections will be:

$$p(i) = \frac{out(i)}{n}$$

This guess has to satisfy $M^T p = p$, we will see that it indeed does:

$$(M^T p)(i) = \sum_{j->i} \frac{p(j)}{out(j)} = \sum_{j->i} \frac{out(j)}{n \cdot out(j)} = \sum_{j->i} \frac{1}{n} = \text{(symmetric graph)} = \sum_{i->n} \frac{1}{n} = \frac{out(i)}{n} = p(i)$$

After proving our guess, we proceeded to check that this immediate solution works properly in a real symmetric graph and that is the same as the one obtained with the power method. In order to do so, we computed the theoretical page rank, using the former expression, and compared it to the page rank obtained by the application of the random surfer algorithm (i.e. power method).

To make a fair comparison, we used a high value of the damping factor (d = 0.99) to modify as little as possible our original graph. The metric of comparison was the cosine similarity, obtaining a value of 0.99779 between the vectors of weights of both solutions.

As this correlation is almost 1, we have proven that our solution is equivalent to the one given by the power method. This way, we get a much faster page-rank algorithm for symmetric graphs (O(n) if we know out(i), compared to O(iterations * |E|)). As |E| >> n, the asymptotic cost of our proposed solution is smaller than the iterative method.

Final Considerations

Overall, this laboratory has helped us to gain more insight about the pagerank computation as well as the different implementations of the random surfer algorithm. Being able to observe, and code, these implementations has allowed us to rethink how the general IR algorithms could be adapted, depending on the structure of our input data, to boost the performance.