Information Retrieval.

Pagerank

Network of airports and flights

Student:
Maria Slanova
DMKM, 2015

Introduction

In this session, we have used text files from Open Flights, containing the airports and flight information, to build a network of airports and flights. We have programed a version of Pagerank on this data. Since "Routes.csv" contained all the necessary information about node and edges, we didn't use "Airports.csv" file for Pagerank computation.

The implementation of Pagerank algorithm is done in Python using the following major packages:

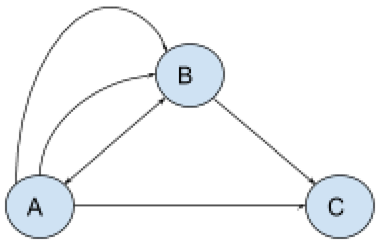
Pandas - http://pandas.pydata.org

Networkx - https://networkx.github.io/documentation/latest/download.html

Preprocessing

In the very beginning we went through the following preprocessing steps:

- deleted all the unnecessary columns, such as airline code, openflights airline code. After this step we had a file, containing only 3 letter IATA code for origin airport and for destination airport. This file was further used as an input for our program.
- using the python library "Pandas", we combined the tuples representing the same routes and added the information about the weights of each edge. Following graph and table show an example of this preprocessing step:



Initial edges:

A	B
A	B
A	B
A	C
B	A
B	C

From node	To Node	Weight
A	B	3
A	C	1
B	A	1
B	C	1

- afterwards we used a nested dictionary to present the weighted graph. The dictionary contains for each node the information on nodes, that point to it and the weight of each link. The dictionary for the previous example would look like following:
  {'A': {'B': 1}, 'C': {'A': 1, 'B': 1}, 'B': {'A': 3}}
- for the computation of Pagerank we need the outdegree value of a node. These values were also represented as a dictionary. For this purpose we used build-in NetworkX function - *nx.DiGraph.out_degree*
  {u'A': 4.0, u'C': 0, u'B': 2.0}

PageRank

The main difficulties in implementing the algorithm were, to add the effect of virtual edges efficiently for the nodes which don't have any outgoing edge. In order to do so we have to see how much pagerank do they add in total to each vertex in particular. For this computation, in each iteration, we sum the pagerank of all the nodes having no outgoing edge, then divide this sum by total number of nodes 'N' and add the resulting value to the pagerank of each node. This helps us creating an effect of outgoing edges from a node which has originally no outgoing edge. Although from this we are also adding an effect of a path from this node to itself, we ignore this effect since its very small probability.
The chosen stopping criteria for the Pagerank is a number of iterations. For 100 iterations it took 5 seconds. The chosen value for damping factor is 0.85.

Conclusions:

- do not use list or matrix representation of the graph. Using those first, made us wait approximately 20 min per iteration.
- initially we used as a threshold the sum of differences between the new and old pagerank values for each node. But even putting this sum equal or less than 10^{-10} resulted in a very small number of iterations (number of iterations $\leq$ 5)