UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

IRRS Lab 3 Report

PageRank Implementation for Flight Routes

Olivia de la Chappelle
MSc student
olivia.de.la.chappelle@estudiantat.upc.edu

Emmanuel Werr
MSc student
emmanuel.werr@estudiantat.upc.edu

Facultat d'Informàtica de Barcelona
Master of Data Science

November 9, 2022

1 Introduction

This lab aimed to understand the PageRank algorithm and implement a version of it in Python using two text files with airports and flight routes to build a network of airports and flights, with the end goal of computing the Page Rank of each airport. Below are our results and difficulties encountered while building the required network and implementing the PageRank algorithm.

2 Reading the Files

We decided to create a "route" class that we used as a step to read the route.txt file. In fact, by gathering each route, with its origin and destination information, in the class "route" it is easy to create the Edge class.

The function called createEdge() allows us to go through the list of routes called "routeList" and find the duplicates. The number of duplicates, for each possible combination of origin and destination, will represent the weight of this connection. At the end of the compilation of the function, we obtain a list of Edges. Each element of the list has an origin, a destination, and a weight.

As we can tell is the class airport, we are supposed to list the different possible routes arriving at the concerned airport. Having created the Edge route corrected, we added the list of routes incoming to the airport. Additionally, we added the out-weight value for each airport, which is the number of edges coming out of the airport.

We also took the following steps to prevent bugs:

- We discarded routes that do not link airports included in our dataset
- We added the out-weight value to each airport
- We stored the edges that arrive at an airport in his own information

The difficulty was mainly to find these small modifications to make the code run properly and efficiently.

3 Implementing the PageRank Algorithm

To implement the computePageRank function we understood the need of the vectors P and Q. They are used to store the page rank of each airport in the graph. P is the original and Q is a copy of P we update in the for loop. At each iteration, we store the new Q in P, and reiterate this logic. When finally, two P's at consecutive iterations are sufficiently close we stop the loop. (When the cosine similarity between the two vectors P and Q is above 0.95).

During the calculation of PageRank in the algorithm, we separated the 2455 airports that do not have incoming or outcoming edges. The issue is that if the random surfer

is ending on one of those nodes in the graph, it will be blocked and will not be able to move to another node, apart when, thanks to the damping factor, the random surfer is placed elsewhere. So to solve this issue, we calculated the total PageRank accumulated by those airports and distributed the value to all of the other airport's PageRank.

To implement the while loop, we decided to make it stop when the global mean value of the absolute value of (P-Q)/P is lower than 0.05. Moreover, we chose a 0.85 damping factor. One of the main purposes of the damping factor is to lessen the negative effects of "sinks" (in our case, airports without any outgoing flights) within our network. It represents the probability that "a person" traveling from airport to airport would keep on traveling. A damping factor of 1 would mean that the person would never stop moving from one airport to the next, so they will inevitably, eventually, end up in a sink. Thus a lower damping factor would introduce a random probability that a person would stop at any given airport. We observed marginally better results while using a standard damping factor of 0.85 (relatively high).

Below is a screenshot of the result on the terminal after launching PageRank.py. We also sent you the "outputPageRank" file in the folder with the results. It is a document that recapitulates the PageRank of each airport in the final graph.

```
Reading Airport file from airports.txt
There were 5740 Airports with IATA code
Reading Routes file from {fd}
There were 68292 Routes with IATA code
The EdgeList was succesfully created in variable edgeList
The AirportList was succesfully filled by its corresponding edges
                  PageRank Airport_name                                 Airport_info
2723   [28.544990884906664]          DEN                  Denver Intl, United States
2801   [26.667826061511175]          ORD              Chicago Ohare Intl, United States
2460     [26.0691178414766]          LAX                Los Angeles Intl, United States
2343   [21.806574065255752]          SYD                        Sydney Intl, Australia
2655   [21.248946516866784]          ATL  Hartsfield Jackson Atlanta Intl, United States
...                     ...          ...                                           ...
2882  [0.36357142857142855]          AMI                    Selaparang, Indonesia
2891  [0.36357142857142855]          NAK              Nakhon Ratchasima, Thailand
2906  [0.36357142857142855]          LPX                     Liepaja Intl, Latvia
2908  [0.36357142857142855]          SQQ                 Siauliai Intl, Lithuania
5739  [0.36357142857142855]          GLI                    Glen Innes, Australia

[5740 rows x 3 columns]
#Iterations: 6
Time of compute PageRanks(): 0.8294870853424072
```

Figure 1: pageRank.py terminal output after running

4 Difficulties and Final Thoughts

This lab presented many difficulties, especially in implementing the PageRank algorithm properly and establishing a network that allowed it to perform the necessary calculations efficiently. At first, we attempted to build different data structures to hold the necessary

information for PageRank computation. Including the use of a transition matrix in order to calculate PageRank iteratively using the power method (which proved to be very inefficient). We experimented with different damping factors and observed how they made the final PageRanks converge faster or slower. We also experimented with different stopping criteria for the while loop within the PageRank computation.

There is still much left to understand about the intricacies of implementing this algorithm at scale but this lab proved to be a rewarding experience filled with many learning opportunities.