Emmanuel Werr & Olivia de la Chapelle

IR-MIRI Lab, Session 3: PageRank

Here is the report talking about the way we implemented the “PageRank” lab. We are going to explain the main difficulties and choices we made to arrive at this conclusion.

1. Reading the files airports.txt and routes.txt

We decided to create a route class that we used as a step to read the route.txt. 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 amount 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 to the concerned airport. Having created the Edge route corrected, we added the list of routes incoming to the airport. Additionally, we added the outweight value for each airport, which is the number of edges coming out of the airport.

We also did those actions to prevent bugs:

- We got rid of routes that do not link airports included in our dataset
- We added the outweight value to each airport
- We stored the edges that arrive to an airport in his own information

The difficulty was mainly to find these small modifications to make to the code to make it run.

2. 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).

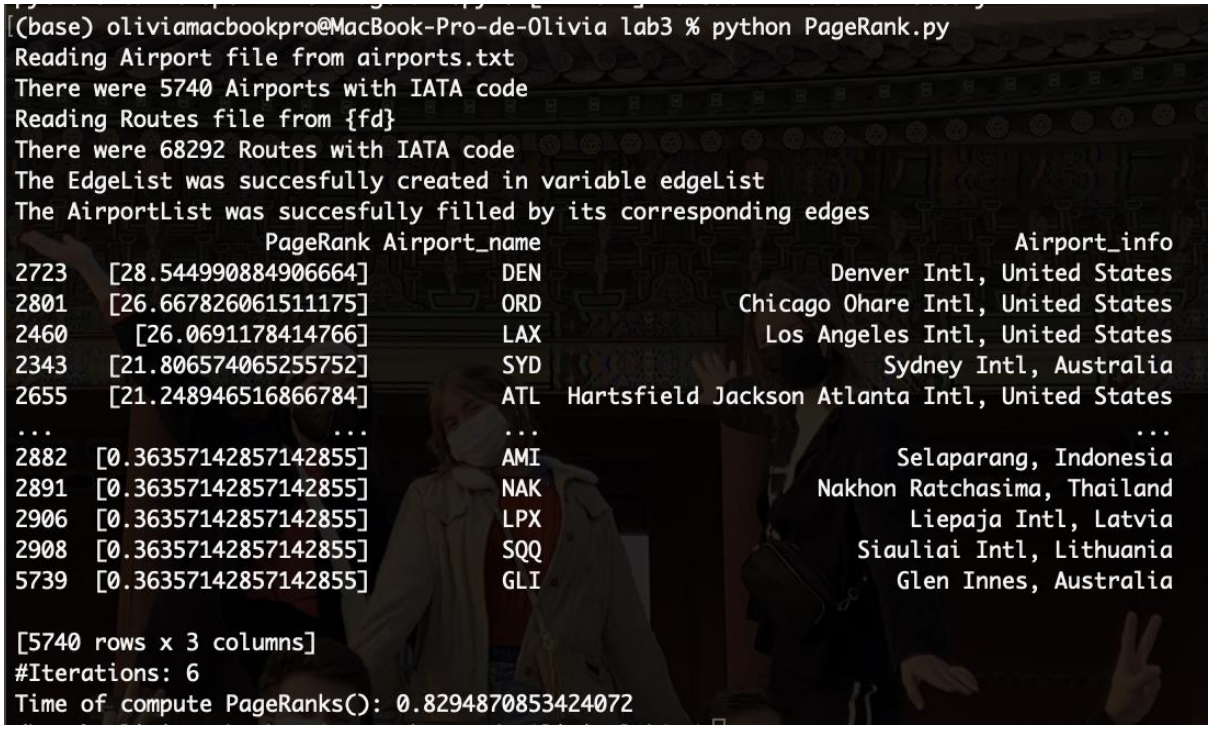
During the calculation of PageRank is the algorithm, we separated the 2455 airports that do not have incoming or outcoming edges. The issue is that if the random surfer is ending on one of those nodes in the graph, it will be blocked and will not be able to move to another node, apart when, thanks to 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 as suggested by Brin and Page because we noticed it was the most common one.

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

We did need 6 iterations of the airports to obtain the final PageRank values.



```
(base) oliviamacbookpro@MacBook-Pro-de-Olivia lab3 % python PageRank.py
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
```

To conclude, this lab was very interesting because it made us work on a topic that is very concrete. Being able to fully understand how the Google search algorithm work is definitely a skill that will be useful in the future as Engineer in Artificial Intelligence.