PR

November 5, 2016

In [190]:
```
# Loading python libraries
from __future__ import division
import csv
import pandas as pd
import numpy as np
import networkx as nx
```

In [191]:
```
# PreProcessing of airports.txt using bash awk:
# Remove double quotes in airports.txt, Reason -> better readability when
# ('"Info"') -> messy, ('Info') -> more readable
```

In [192]:
```
# List of airports with only IATA label
airports = []
with open('/home/kopsor/FIB/IR/lab3/airport.txt') as file:
    for line in file:
        a = line.split(',')
        airports.append(a[4])

# We clear the airports of empty IATA codes,then order them
# and finally map the airport codes with integers in a map

airports = filter(lambda a: a != '', airports)
airports.sort()
airports = dict((air, airports.index(air)) for air in airports)
air = range(0,len(airports))
```

In [193]:
```
# Put the routes into list of tuples where tuple
# looks like this (source, destination)
# and we mapped the IATA string to integers
# so we have (int, int) instead of (string, string)
# we did this because of better operations with graph data strcture

routes = []
incomplete = 0
with open('/home/kopsor/FIB/IR/lab3/routes.txt') as file:
    for line in file:
        r = line.split(',')
        try:
```

```
                source = airports[r[2]]
                destination = airports[r[4]]
                routes.append((source,destination))
            except:
                incomplete += 1

    # Some of the routes were rejected, because IATA codes
    # from routes file were not present in airport file
    print incomplete, "Routes rejected"
```

```
528 Routes rejected
```

In [194]:
```
    # We used directed graph

    G = nx.DiGraph()
    G.add_nodes_from(air)
    G.add_edges_from(routes,weight=0)

    # Calculate weight for each edge
    # weight is the number of flights from the same src to same dst
    for route in routes:
        G[route[0]][route[1]]["weight"] += 1

    # Calculate the sum of weights of outgoing edges for each node
    n = len(airports)
    out = np.zeros(n)
    for i in range(n):
        out_edges = G.out_edges(i)
        if not out_edges:
            continue
        # i - source of edge edge[1] - destination of edge
        out[i] = sum(G[i][edge[1]]["weight"] for edge in out_edges)
```

In [197]:
```
    # Here we have the pagerank algorithm
    # at first we try to do it with n*n matrix
    # but it took 20 - 30 mins :D but then we have optimized it
    # with graph (takes only 5 - 10 seconds)

    iters = 0
    error_limit = 10**-8
    n = len(airports)
    P = np.ones(n)/n
    L = 0.85
    converged = False

    print "Sum of P before ", sum(P)
```

```
        while not converged:
            iters += 1
            Q = np.zeros(n)
            for i in range(0,n):
                in_edges = G.in_edges(i)
                if not in_edges:
                    Q[i] = P[i]
                    continue
                destinations = [destination[0] for destination in in_edges]
                Q[i] = L * sum(P[j] * G[j][i]["weight"] / out[j]
                               for j in destinations
                               if G.has_edge(j,i)) + (1-L)/n

            error = np.linalg.norm(Q - P)
            if (error <= (n * error_limit)):
                converged = True

            P = Q


        print "Number of iterations ", iters
        print "Sum of P after ", sum(P)
```

```
Sum of P before  1.0
Number of iterations  15
Sum of P after  1.00385700325
```

In [198]:
```
        # Here we create list of python dictionaries,
        # in each dict there is info about certain airfield

        airports_info = []
        with open('/home/kopsor/FIB/IR/lab3/airport.txt') as file:
            for line in file:
                a = line.split(',')
                airports_info.append({"pagerank":0, "IATA":a[4], "airport":a[1] ,
```

In [199]:
```
        # We map the pagerank score to specific airfields

        for a in airports_info:
            try:
                score = P[airports[a["IATA"]]]
                a["pagerank"] = score
            except:
                pass
```

In [201]:
```
        # We put the results into pandas dataframe for better viewing of results

        df = pd.DataFrame(airports_info)
```

```
df = df.sort(["pagerank"], ascending=[0])
df.head()
```

```
/home/kopsor/.local/lib/python2.7/site-packages/ipykernel/__main__.py:4: FutureWarn
```

Out[201]:

	IATA	airport	city	country	pagerank
3385	LAX	Los Angeles Intl	Los Angeles	United States	0.003579
3731	ORD	Chicago Ohare Intl	Chicago	United States	0.003570
3652	DEN	Denver Intl	Denver	United States	0.003557
503	LHR	Heathrow	London	United Kingdom	0.002787
3222	SIN	Changi Intl	Singapore	Singapore	0.002753

In [202]:
```
# Here we compare results of our pagerank implementation
# with networkx's implementation

pr = nx.pagerank(G, alpha=0.85)
for a in airports_info:
    try:
        score = pr[airports[a["IATA"]]]
        a["pagerank"] = score
    except:
        pass
df = pd.DataFrame(airports_info)
df = df.sort(["pagerank"], ascending=[0])
df.head()
```

```
/home/kopsor/.local/lib/python2.7/site-packages/ipykernel/__main__.py:12: FutureWar
```

Out[202]:

	IATA	airport	city	country	pagerank
3385	LAX	Los Angeles Intl	Los Angeles	United States	0.005595
3731	ORD	Chicago Ohare Intl	Chicago	United States	0.005565
3652	DEN	Denver Intl	Denver	United States	0.005555
503	LHR	Heathrow	London	United Kingdom	0.004330
3583	ATL	Hartsfield Jackson Atlanta Intl	Atlanta	United States	0.004270