TME5

April 9, 2019

1 PageRank Implementation

```
In [ ]: liste_nodes = open("/Vrac/CPA-PageRank/alr21--pageNum2Name--enwiki-20071018.txt", "r",e
        map_sommet_page = dict()

        for e in liste_nodes:
            if e.startswith("#") or e.startswith("\n"):
                continue
            e = e.rstrip("\n")
            indice_point_1 = e.split("\t")

            map_sommet_page[int(indice_point_1[0])] = indice_point_1[1]

        liste_nodes.close()
```

1.1 Chargement du graphe de sites web:

```
In [ ]: # Structure de données: Liste d'adjacence
        cleaned_graph = open("/Vrac/CPA-PageRank/alr21--dirLinks--enwiki-20071018.txt", "r",en

        max_int = -1
        for e in cleaned_graph:
            if e.startswith("#"):
                continue
            e = e.rstrip("\n")
            indice_point = e.split("\t")
            if int(indice_point[0])>max_int:
                max_int = int(indice_point[0])
            if int(indice_point[1])>max_int:
                max_int = int(indice_point[1])
        cleaned_graph.close()

        cleaned_graph = open("/Vrac/CPA-PageRank/alr21--dirLinks--enwiki-20071018.txt", "r",en

        map_sommet_succ = dict()
        for i in range(max_int+1):
            map_sommet_succ[i] = [[],0,0]
```

```
        map_sommet_pred = dict()
        for i in range(max_int+1):
            map_sommet_pred[i] = [[],0]

        for e in cleaned_graph:
            if e.startswith("#") or e.startswith("\n"):
                continue
            e = e.rstrip("\n")
            indice_point_1 = e.split("\t")
            map_sommet_succ[int(indice_point_1[0])][0].append(indice_point_1[1])
            map_sommet_succ[int(indice_point_1[0])][1] = map_sommet_succ[int(indice_point_1[0])
            map_sommet_succ[int(indice_point_1[0])][2] = 1/map_sommet_succ[int(indice_point_1[0

            map_sommet_pred[int(indice_point_1[1])][0].append(indice_point_1[0])
            map_sommet_pred[int(indice_point_1[1])][1] = map_sommet_succ[int(indice_point_1[1])

        cleaned_graph.close()
```

In [ ]:
```
map_sommet_succ_copy = dict()
for k,v in map_sommet_succ.items():
    if v[0] != []:
        map_sommet_succ_copy[k] = v
```

1.1.1 Produit Matriciel

In [ ]:
```
print(len(list(map_sommet_succ.keys())))
def norm(p):
    #print(p)
    res = 0
    for i in list(p.keys()):
        res = res + p[i]
    return res
```

In [ ]:
```
def normalize(p,n,norma):
    res = p
    i = 0
    for i in list(p.keys()):
        normalisated = (1 - norma)/n
        res[i] =  res[i] + normalisated
    return res
```

In [ ]:
```
import copy

def compare(p,vec,epsilon):
    keys = list(p.keys())
    for i in keys:
        #print(abs(p[i]-vec[i]), " epsilon = ",epsilon )
        if (abs(p[i]-vec[i])>epsilon):
```

```
                return False

        return True
```

In [ ]:
```
def power_iteration_1(liste_succ,alpha,n):
    vec = dict()
    keys = list(liste_succ.keys())
    p = dict()
    for i in keys:
        vec[int(i)] = 1/n
        p[i] = 0
    d = 1 - alpha


    j = 0
    while(True):


        for i in keys:
            if liste_succ[i][0] == []:
                continue
            for v in liste_succ[i][0]:
                v = int(v)
                #print(" valeur =",(d*vec[i]/liste_succ[i][1]), "p[v] = ",p[v])
                p[v] = p[v] + (d*vec[i]/liste_succ[i][1])

        print("boucle for")
        norma = norm(p)
        p = normalize(p,n,norma)

        j = j + 1
        print(j)
        if (j==15):
            break

        for i in keys:
            if liste_succ[i][0] == []:
                continue
            vec[i] = p[i]
            p[i] = 0

    return p
```

In [ ]:
```
p = power_iteration_1(map_sommet_succ,0.15,len(list(map_sommet_succ.keys())))
```

In [ ]:
```
p2 = power_iteration_1(map_sommet_succ,0.9,len(list(map_sommet_succ.keys())))
```

In [ ]:
```
p1 = copy.deepcopy(p)
```

```
In [ ]: def getKeyByValue(p,value):
            key = 0
            for k,v in p.items():
                if (value == v):
                    key = k
            return key

        def getHighestPageRanks(p):
            l = list()
            for i in range(5):
                pmax = max(list(p.values()))
                indice_max = getKeyByValue(p,pmax)
                l.append(indice_max)
                del(p[indice_max])
            return l

        def getLowestPageRanks(p):
            l = list()
            for i in range(5):
                pmin = min(list(p.values()))
                indice_min = getKeyByValue(p,pmin)
                l.append(indice_min)
                print(indice_min)
                print(pmin)
                print("---------")
                del(p[indice_min])
            return l
```

```
In [ ]: id_highest_page_ranks = getHighestPageRanks(p1)
```

```
In [ ]: highest_page_ranks = list()
        for i in id_highest_page_ranks:
            highest_page_ranks.append(map_sommet_page[i])
```

```
In [ ]: highest_page_ranks
```

```
In [ ]: id_lowest_page_ranks = getLowestPageRanks(p1)
```

```
In [ ]: lowest_page_ranks = list()
        for i in id_lowest_page_ranks:
            print(map_sommet_page[i])
            lowest_page_ranks.append(map_sommet_page[i])
```

```
In [ ]: lowest_page_ranks
```

1.2 Exercice 2: Correlations

```
In [ ]: y = []
        for i in list(p.keys()):
            y.append(map_sommet_pred[i][1])
```

```
In [ ]: import matplotlib.pyplot as plt
        plt.figure()
        plt.plot(list(p.values()),y,'ro')
        plt.xscale('log')

In [ ]: plt.show()
```