databricks Page Rank 2023-10-27 15:05:09

(https://databricks.com)

```
%python
pip install networkx
```

```
Python interpreter will be restarted.
Collecting networkx
  Downloading networkx-3.2-py3-none-any.whl (1.6 MB)
Installing collected packages: networkx
Successfully installed networkx-3.2
Python interpreter will be restarted.
```

```
from pylab import rcParams
rcParams['figure.figsize']= (3,3)


def nice_print(v , digits = 3):
    format = '%%.%df' %digits
    print(', '.join([format % e for e in v]))


nice_print([.12333122, .13432221, .64442143])
nice_print([.12333122, .13432221, .64442143], digits = 4)
```

```
0.123, 0.134, 0.644
0.1233, 0.1343, 0.6444
```

```
labels = ['A',
          'B',
          'C',
          'D',
          'E',
          'F',
          'G']
pages = range(len(labels))

positions = [(0,1),
             (0,2),
             (2,2),
             (0,0),
             (1,0),
             (2,0),
             (1,1)]

# this dictionary accosciates the numbers in pages to labels
page_labels = {p: l for p, l in zip(pages, labels)}
page_labels
```

```
Out[10]: {0: 'A', 1: 'B', 2: 'C', 3: 'D', 4: 'E', 5: 'F', 6: 'G'}
```

```
links = [(1,0),
             (3,0),
             (0,1),
             (5,2),
             (6,2),
             (6,5),
             (5,6),
             (2,6),
             (0,6),
             (5,4),
             (4,3)]
```

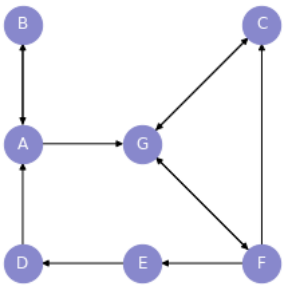
```
import networkx as nx
import matplotlib.pyplot as plt

g = nx.DiGraph()

for p in pages:
    node = g.add_node(p)

for (a,b) in links:
    g.add_edge(pages[a], pages[b])
```

```
plt.clf()
display(nx.draw(g, with_labels = True, labels = page_labels, node_size = 800, node_color = '#8888CC', font_color = 'white', pos
= positions))
```




```
adjacency ={}
for u in range(len(pages)):
    adjacency[u] = []

for (a,b) in links:
    adjacency[a].append(b)

print(adjacency)
```

```
{0: [1, 6], 1: [0], 2: [6], 3: [0], 4: [3], 5: [2, 6, 4], 6: [2, 5]}
```

```
connection_matrix =[]
for a in adjacency:
    for b in adjacency[a]:
        connection_matrix.append((b,a,1./len(adjacency[a])))
connection_matrix
```

```
Out[17]: [(1, 0, 0.5),
 (6, 0, 0.5),
 (0, 1, 1.0),
 (6, 2, 1.0),
 (0, 3, 1.0),
 (3, 4, 1.0),
 (2, 5, 0.3333333333333333),
 (6, 5, 0.3333333333333333),
 (4, 5, 0.3333333333333333),
 (2, 6, 0.5),
 (5, 6, 0.5)]
```

```
links_RDD = sc.parallelize(connection_matrix).cache()
```

```
links_RDD.take(3)
```

```
Out[19]: [(1, 0, 0.5), (6, 0, 0.5), (0, 1, 1.0)]
```

```
import numpy as np
n = len(pages)
page_rank = np.ones(n)/n
old_page_rank = np.ones(n)
print("Page rank is", page_rank)
print("Old Page rank is", old_page_rank)
```

```
Page rank is [0.14285714 0.14285714 0.14285714 0.14285714 0.14285714 0.14285714
 0.14285714]
Old Page rank is [1. 1. 1. 1. 1. 1. 1.]
```

```
def l2distance(v,q):
    if len(v) != len(q):
        raise ValueError('Cannot compute the distance of two vectors of different size')
    return sum([(q_el-v_el)**2 for v_el, q_el in zip(v,q)])
```

```
tolerance = 10e-7
max_iterations = 1000
iteration = 0
print("iiiiiiii")
while(l2distance(old_page_rank, page_rank) >= tolerance and iteration < max_iterations):
    old_page_rank = page_rank
    page_rank_values  = links_RDD.map(lambda x:(x[0], x[2]*page_rank[x[1]])).reduceByKey(lambda a, b: a+b).sortByKey().collect()
    page_rank = np.array([c for (i,c) in page_rank_values])
    nice_print(page_rank)
    print("Page rank", page_rank)
    iteration += 1
```

```
iiiiiiii
```