데이터 과학

L15.1: PageRank Practice

Kookmin University

모듈 불러오기

- 사용할 모듈 import 하기

```
import matplotlib.pyplot as plt
import numpy as np
import requests
```

데이터 준비

- 하이퍼링크 그래프 다운로드 받기
  - example_index: 노드 정보
  - example_arc: 에지 정보

```
# http://webdatacommons.org/hyperlinkgraph

with open("example_index", "wb") as f:
   r = requests.get("http://webdatacommons.org/hyperlinkgraph/data/example_index")
   f.write(r.content)

with open("example_arcs", "wb") as f:
   r = requests.get("http://webdatacommons.org/hyperlinkgraph/data/example_arcs")
   f.write(r.content)
```

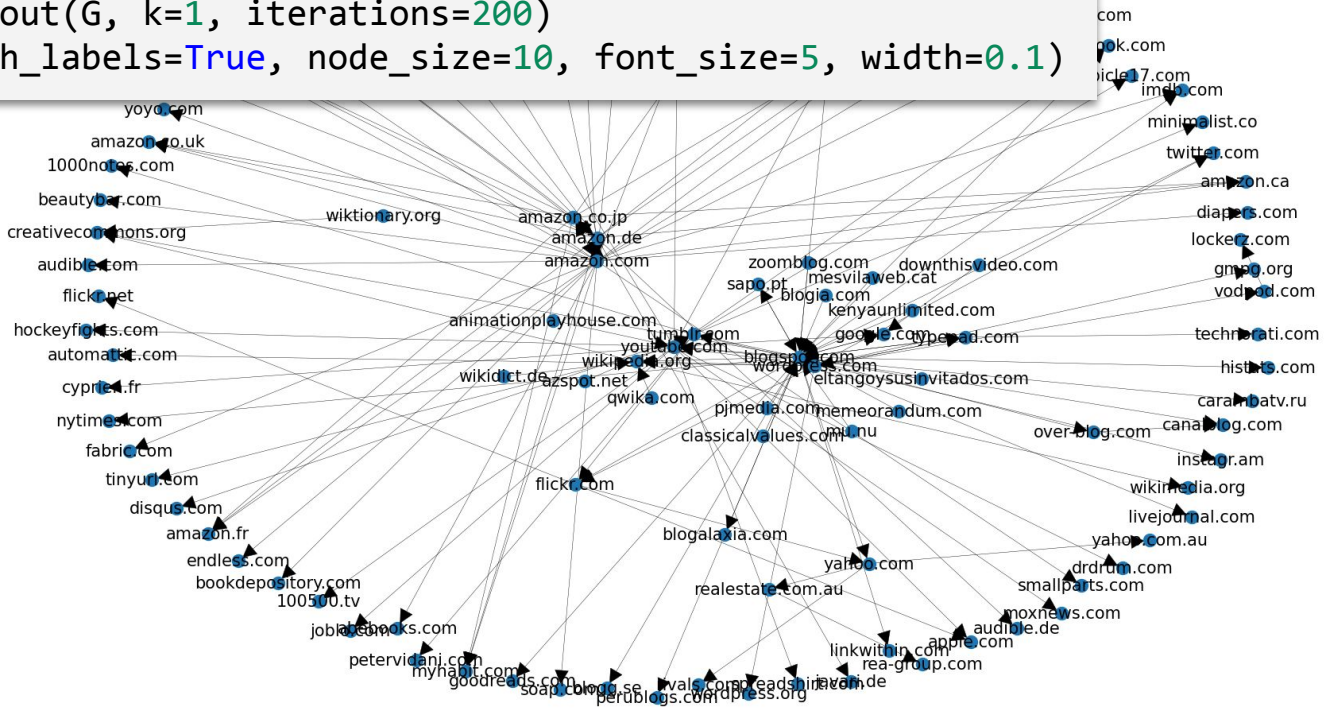
데이터 준비

- 하이퍼링크 그래프 불러오기

```
nodes = np.loadtxt("example_index", dtype=object)[:,0]
num_nodes = nodes.shape[0]

edges = np.loadtxt("example_arcs", dtype=int)
num_edges = edges.shape[0]
```

데이터 살펴보기

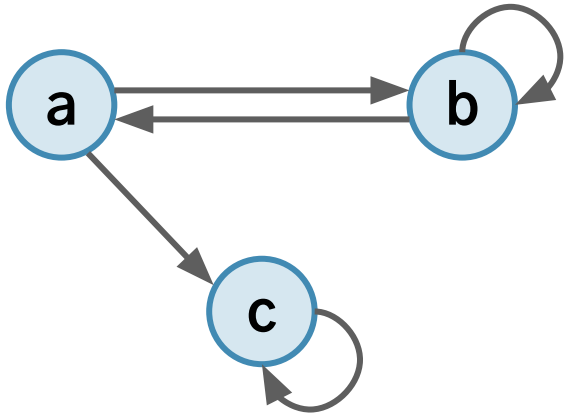
```
import networkx as nx
import matplotlib.pyplot as plt
plt.figure(dpi=300)
G = nx.DiGraph()

for e in edges:
   G.add_edge(nodes[e[0]], nodes[e[1]])

pos = nx.spring_layout(G, k=1, iterations=200)
nx.draw(G, pos, with_labels=True, node_size=10, font_size=5, width=0.1)
```



Google Matrix

- Google Matrix는 Dense 함
  ⇒ 연산이 과도하게 많음

$$A = \beta\, M + (1-\beta)\left[\frac{1}{N}\right]_{N\times N}$$



$$0.8\times \begin{bmatrix} 0 & 1/2 & 0 \\ 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \end{bmatrix} +0.2\times \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

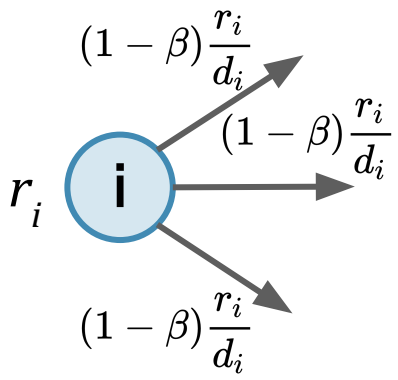
M $\qquad$ $[1/N]_{N\times N}$

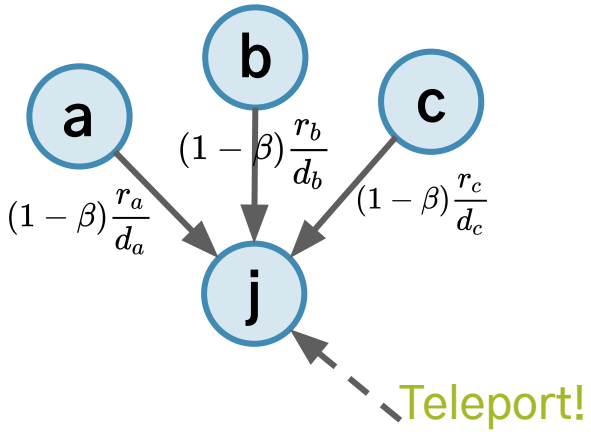
$$= \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

A

보다 효율적인 구현 방법

- Adjacency List 형식으로 데이터를 구성
- 각 노드마다, 그 노드의 점수를 out neighbor들에 골고루 나눠줌
- 각 노드마다, 받은 점수를 합산 함





pagerank()

- edges: 에지 목록
- beta: teleport 확률
- threshold: r 벡터 변화량이 이 값보다 작을경우 계산 중단
- epochs: 최대 반복 계산 횟수

```
def pagerank(edges, beta, threshold=10^-20, epochs=100):

    # Adjacency List 만들기
    neighbors = [[] for _ in range(num_nodes)]

    for e in edges:
        neighbors[e[0]].append(e[1])

    # 벡터 초기화: 모든 값을 (1/n)으로 설정
    r = [1/num_nodes] * num_nodes
```

pagerank()

```
for epoch in range(epochs):

    # 업데이트된 값을 저장할 벡터 초기화
    r_next = [0] * num_nodes

    # 모든 노드에서...
    for u in range(num_nodes):
        # 그 노드가 가리키는 이웃 노드에게...
        for v in neighbors[u]:
            # 현재 자신의 점수 r[u]에 (1-beta)을 곱하고, 등분하여 나눠줌
            r_next[v] += (1 - beta) * r[u] / len(neighbors[u])

    # 1-(r_next의 값들의 합)은 teleport에 의해 각 노드에 도착할 확률임
    # 그러므로, 1-(r_next의 값들의 합)을 모든 노드에게 등분하여 나눠줌
    teleport_prob = 1-sum(r_next)
    for u in range(num_nodes):
        r_next[u] += teleport_prob/num_nodes

    # 이전 벡터 r과 새로운 벡터 r_next 사이의 변화량을 합산
    delta = sum(abs(a-b) for a, b in zip(r, r_next))

    # 원래 벡터를 새로 계산한 벡터로 교체
    r = r_next

    if delta < threshold:
        break

return r
```

계산 및 출력

```
r = pagerank(edges, 0.15)
for score, node in sorted(zip(r, nodes), reverse=True)[:10]:
   print(f"({score:.6f}) {node}")
```

```
(0.085426) blogspot.com
(0.024220) creativecommons.org
(0.021582) wikipedia.org
(0.017731) canalblog.com
(0.016119) youtube.com
(0.015904) tumblr.com
(0.015904) google.com
(0.015600) wikimedia.org
(0.015058) rea-group.com
(0.013151) yahoo.com
```

Questions?