cs224w colab 0

1. NetworkX Tutorial

- directed, undirected, multigraph 와 같은 다양한 그래프 타입들 제공
- Graph

```
import networkx as nx

G=nx.Graph() ## Undirected graph
H=nx.DiGraph() ## Directed graph

G.is_directed() ## 이것을 통해 directed graph 여부 확인
G.graph ## 이것을 통해 graph level attribute 확인

G.graph['Name']='Bar' ## graph level attribute 삽입
```

- Node

```
## 새로운 노드 하나 추가
G.add_node(0,feature=5,label=0)
node_0_attr=G.nodes[0] ## 노드 0의 속성 출력

## 노드 여러 개 추가
G.add_nodes_from([
  (1, {"feature": 1, "label": 1}),
  (2, {"feature": 2, "label": 2})
]) #(노드, dictionary 형태의 속성들)

for node in G.nodes(): ## 해당 graph에 존재하는 node들 반환
  print(node)


for node in G.nodes(data=True): ## data=True라고 할 경우 node들 속성 반환
  print(node)

# graph가 가진 노드들의 개수 반환
num_nodes = G.number_of_nodes()
print("G has {} nodes".format(num_nodes))
```

- Edge

```
## undirected graph에 새로운 edge 하나 추가
G.add_edge(0,1,weight=0.5) ## 이때 가중치는 0.5
```

```
## undirected graph에 여러개 edge 추가
G.add_edges_from([
  (1, 2, {"weight": 0.3}),
  (2, 0, {"weight": 0.1})
])

for edge in G.edges(): ## 해당 graph에 존재하는 edge들 반환
  print(edge)

## edge 속성 확인 - 이때 undirected graph라 (0,1)==(1,0)
edge_0_1_attr=G.edges[(0,1)]
edge_1_0_attr=G.edges[(1,0)]

# edge 개수 확인
num_edges = G.number_of_edges()

## directed graph에 새로운 edge 하나 추가
H.add_edge(0,1,weight=0.5)

## edge 속성 확인 - 이때 directed graph라 (1,0)에 대해 속성 출력하면 에러남
edge_0_1_attr=H.edges[(0,1)]
edge_1_0_attr=H.edges[(1,0)]
```

- Visualization
  - 간단한 방법

```
nx.draw(G,with_labels=True)
```

  - 다른 방법

```
import matplotlib.pyplot as plt

## 모든 node들에 대한 position - seed는 reporducibility를 위한 것
pos=nx.spring_layout(G,seed=7)

## node 그리기
nx.draw_networkx_nodes(G,pos,node_size=700)

## edge 그리기
nx.draw_networkx_edges(G,pos,edgelist=G.edges(),width=6,edge_color='b',style='dashed')

## node label
nx.draw_networkx_labels(G,pos,font_size=20)

## edge weight label
nx.draw_networkx_labels(G,pos,font_size=20)
edge_labels=nx.get_edge_attributes(G,'weight')
nx.draw_networkx_edge_labels(G,pos,edge_labels)

ax=plt.gca()
ax.margins(0.08)
plt.axis("off")
```

```
plt.tight_layout()
plt.show()
```

- Node degree와 neigbor 찾기

```
## node degree 반환
G.degree[node_id]

## node의 neighbor 반환
list(G.neighbors(node_id)) ## list형 반환

for neighbor in G.neighbors(node_id): ## dict_keyiterator로 반환한다
  print(neighbor)
```

- Other functionalities

```
num_nodes = 4

# 새로운 그래프 생성 후 directed graph로 지정
## path_graph(n)은 n개의 node + n-1개 edge 를 가진 linearly connected graph를 생성한다
G = nx.DiGraph(nx.path_graph(num_nodes))
nx.draw(G, with_labels = True)

# Get the PageRank
pr = nx.pagerank(G, alpha=0.8) ## alpha : 감폭 비율
pr
```

  - PageRank

    incoming link에 대한 structure을 기반으로 graph의 노드들 ranking을 계산한 것