BU.330.760 Deep Learning with Unstructured Data

Lab 2. networkX Drawing and Network Properties

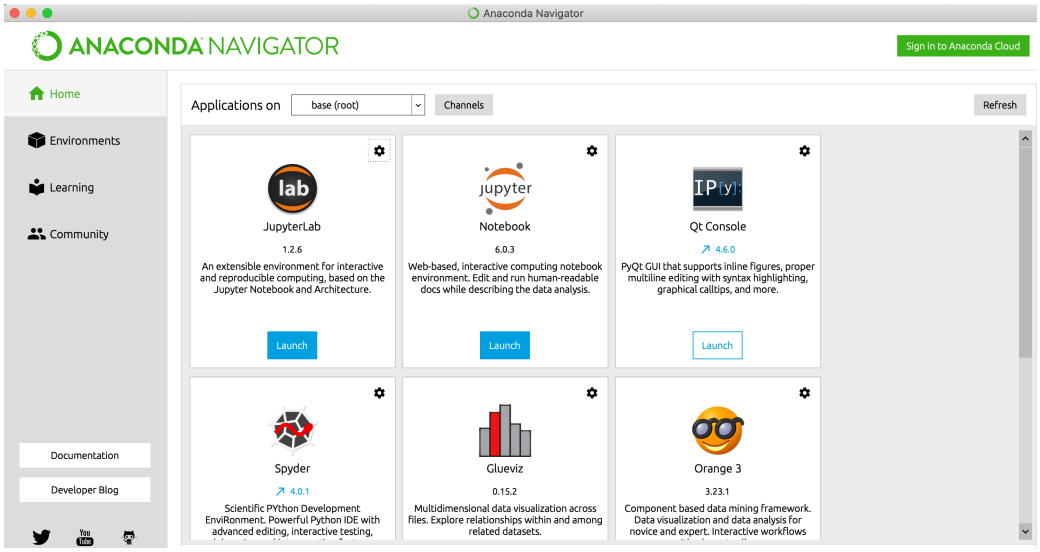
Learning Goal: practice using networkX package to draw a graph and analyze network properties
Required Skills: knowledge on network properties, python Matplotlib

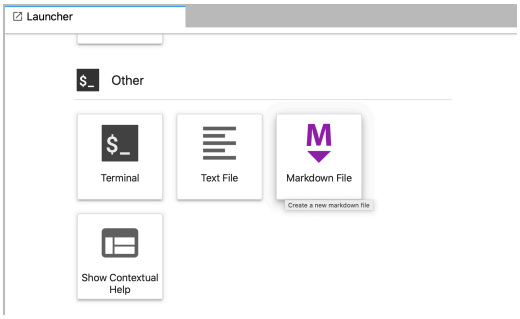
1. We will use networkX (https://networkx.github.io) in lab 2, which is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.
2. For lab 2 to lab 6, I suggest to use Anaconda, a popular Python/R data science platform which provides support for all the packages we will be using including networkX, graph-tool, Keras, Tensorflow and Theano. Download and install Anaconda from https://www.anaconda.com/distribution/ according to your own operating system.

   It is recommended by Anaconda Documentation to install Anaconda for the local user, which does not require administrator permissions and is the most robust type of installation. But some Windows users experienced issues, and please consider installing Anaconda using Administrator account if you are on Windows system. For detailed installation instructions, please refer to https://docs.anaconda.com/anaconda/install/.

3. After you install Anaconda, install networkX package with conda. You can use Terminal on Mac system, or on Windows system, **Aanconda Navigator > JupyterLab > Launcher > Terminal**. Run:

   ```
   conda install -c anaconda networkx
   ```





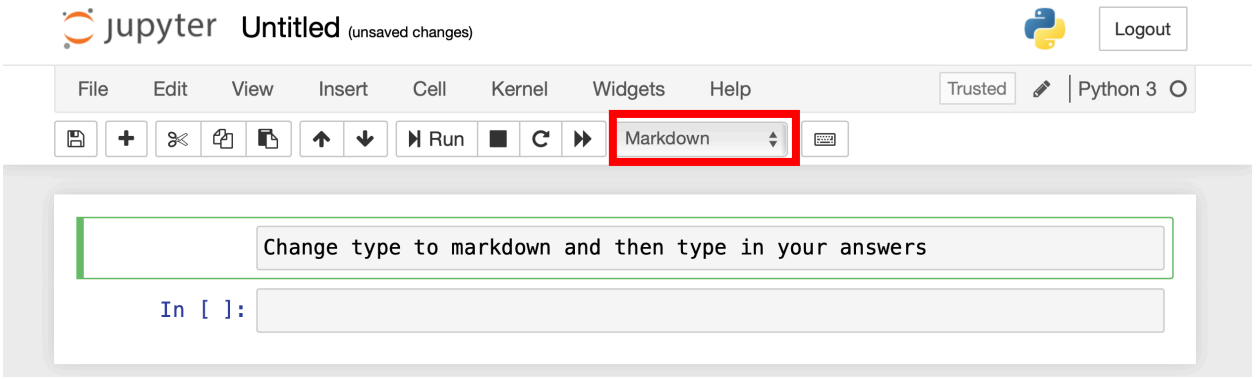
All the screenshots in lab instructions will be using **Anaconda Navigator > Jupyter Notebook** (iPython notebook).

4. If you prefer to use your own Python developer tools or platforms, please install networkX package according to networkX documentation: https://networkx.github.io/documentation/stable/install.html.
5. In Jupyter notebook, or your own Python development environment, build the following scripts
    a. Import required packages.

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

    b. Try the following four basic types of graphs, with different node colors, edge colors, or layout: *path, cycle, complete, star.* In a markdown cell (shown below), type in your answers to

    Q1: Use one sentence each to briefly describe the characteristics of each graph type (its shape, edges, etc..)




```
G = nx.path_graph(5)
nx.draw(G, with_labels=True)
plt.show()

G = nx.cycle_graph(5)
nx.draw(G, node_color='green', with_labels=True)
plt.show()
```

```
G = nx.complete_graph(5)
nx.draw(G, node_color='#A0CBE2', edge_color='red', width=2,
with_labels=False)
plt.show()

G = nx.star_graph(5)
pos=nx.spring_layout(G)
nx.draw(G, pos, with_labels=True)
plt.show()
```

c. We will use lollipop shape graph to test the network properties. Lollipop graph has a head and a stick, and the number of nodes in lollipop head and that of nodes in lollipop stick are specified separately.

```
G = nx.lollipop_graph(3,2)
nx.draw(G, with_labels=True)
plt.show()
```

d. Find the connected component(s) of the graph.

```
list(nx.connected_components(G))
```

e. Calculate the local clustering coefficient of each node.

```
nx.clustering(G)
```

in a markdown cell, answer:

Q2: How many connected components are there in the graph? What are they?

Q3: Which nodes have the highest local clustering coefficient? Explain (from the definition) why they have high clustering coefficient.

f. Calculate these network parameters: radius (which is the shortest eccentricity), diameter, eccentricity, center, periphery and network density.

```
print("radius: %d" % nx.radius(G))
print("diameter: %d" % nx.diameter(G))
print("eccentricity: %s" % nx.eccentricity(G))
print("center: %s" % nx.center(G))
print("periphery: %s" % nx.periphery(G))
print("density: %s" % nx.density(G))
```

g. Now, calculate the betweenness centrality, degree centrality, closeness centrality, and eigenvector centrality for the graph.

```
print("Betweenness")
b = nx.betweenness_centrality(G)
for v in G.nodes():
    print("%0.2d %5.3f" % (v, b[v]))

print("Degree centrality")
```

```
d = nx.degree_centrality(G)
for v in G.nodes():
    print("%0.2d %5.3f" % (v, d[v]))

print("Closeness centrality")
c = nx.closeness_centrality(G)
for v in G.nodes():
    print("%0.2d %5.3f" % (v, c[v]))

print("Eigenvector centrality")
centrality = nx.eigenvector_centrality(G)
sorted((v, '{:0.2f}'.format(c)) for v, c in centrality.items())
```

in a markdown cell, answer:

Q4: Which node(s) has the highest betweenness, degree, closeness, eigenvector centrality? Explain using the definitions and graph structures.

h. Now let's calculate the shortest path length for each pair of nodes, the average shortest path length, and histogram of path lengths.

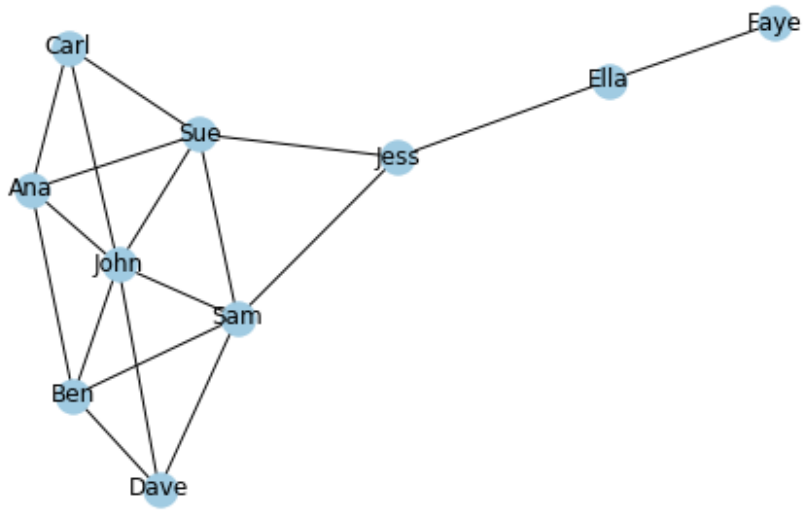
```
pathlengths = []
print("source vertex {target:length, }")
for v in G.nodes():
    spl = dict(nx.single_source_shortest_path_length(G, v))
    print('{} {} '.format(v, spl))
    for p in spl:
        pathlengths.append(spl[p])
print('')
print("average shortest path length %s" % (sum(pathlengths) /
len(pathlengths)))

dist = {}
for p in pathlengths:
    if p in dist:
        dist[p] += 1
    else:
        dist[p] = 1
print('')
print("length #paths")
verts = dist.keys()
for d in sorted(verts):
    print('%s %d' % (d, dist[d]))
```

i. Finally, let's try to relabel the graph nodes with names and plot it.

```
mapping = {0: 'a', 1: 'b', 2: 'c', 3: 'd', 4: 'e'}
H = nx.relabel_nodes(G, mapping)
nx.draw(H, with_labels=True)
plt.show()
```

6. Assignment question: use the kite graph example in lecture 2 slides page 11-13 to calculate the network properties. First draw the kite graph in the following format



Hint 1: use networkX function krackhardt_kite_graph() to create the kite graph.
Hint 2: you need to change the node labels with the names in the graph above. Feel free to use your own node/edge colors.

Secondly, calculate the betweenness, degree, closeness, and eigenvector centrality for each person. Your output should be in the format of "name: centrality score", for example, *John: 0.102*. Feel free to compare your answers with our lecture.

Hint 3: you should be able to fetch the names directly after you relabel the nodes in the previous step. But be careful about the print format.

Submission:
Finish all the steps in the lab, save it to lab2.ipynb file with all the outputs, as well as your answers to Q1-Q4 in markdown cells. Label each markdown cell with question number, such as *Q1: your answer*.
Complete the assignment question, save it to assignment2.ipynb file with all the outputs.
Your submission should include

1. lab2.ipynb and assignment2.ipynb, if you use Jupyter notebook
2. lab2.py and assignment2.py, together with all (labeled) outputs and a plain text file with all the answers, if you use other Python development tools.

Due: Feb.8th 11:30 am EST

Reference:
networkX 2.4 Documentation