Assignment 1

May 17, 2021

---

*You are currently looking at **version 1.1** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ course resource.*

---

1 Assignment 1 - Creating and Manipulating Graphs

Eight employees at a small company were asked to choose 3 movies that they would most enjoy watching for the upcoming company movie night. These choices are stored in the file `Employee_Movie_Choices.txt`.

A second file, `Employee_Relationships.txt`, has data on the relationships between different coworkers.

The relationship score has value of `-100` (Enemies) to `+100` (Best Friends). A value of zero means the two employees haven't interacted or are indifferent.

Both files are tab delimited.

```
In [151]: import networkx as nx
          import pandas as pd
          import numpy as np
          from networkx.algorithms import bipartite


          # This is the set of employees
          employees = set(['Pablo',
                           'Lee',
                           'Georgia',
                           'Vincent',
                           'Andy',
                           'Frida',
                           'Joan',
                           'Claude'])

          # This is the set of movies
          movies = set(['The Shawshank Redemption',
                        'Forrest Gump',
              'The Matrix',
              'Anaconda',
              'The Social Network',
              'The Godfather',
              'Monty Python and the Holy Grail',
              'Snakes on a Plane',
              'Kung Fu Panda',
              'The Dark Knight',
              'Mean Girls'])


# you can use the following function to plot graphs
# make sure to comment it out before submitting to the autograder
def plot_graph(G, weight_name=None):
    '''
    G: a networkx G
    weight_name: name of the attribute for plotting edge weights (if G is weighted)
    '''
    %matplotlib notebook
    import matplotlib.pyplot as plt

    plt.figure()
    pos = nx.spring_layout(G)
    edges = G.edges()
    weights = None

    if weight_name:
        weights = [int(G[u][v][weight_name]) for u,v in edges]
        labels = nx.get_edge_attributes(G,weight_name)
        nx.draw_networkx_edge_labels(G,pos,edge_labels=labels)
        nx.draw_networkx(G, pos, edges=edges, width=weights);
    else:
        nx.draw_networkx(G, pos, edges=edges);
```

1.0.1 Question 1

Using NetworkX, load in the bipartite graph from `Employee_Movie_Choices.txt` and return that graph.

This function should return a networkx graph with 19 nodes and 24 edges

```
In [ ]: def answer_one():

            # Your Code Here
            with open('Employee_Movie_Choices.txt') as f:
                data = f.read()
            #df = pd.DataFrame(data)
            df = pd.read_csv('Employee_Movie_Choices.txt', delimiter = "\t")
            G = nx.from_pandas_dataframe(df, '#Employee', 'Movie')
            all_edges = list(zip(df['#Employee'],df['Movie']))
            #nx.set_node_attributes(G,values = node_attribute_dict,name='node_type')
            B = nx.Graph()
            B.add_nodes_from(employees)
            B.add_nodes_from(movies)
            #B.add_edges_from([('Andy','Anaconda'),('Andy','Mean Girls'),('Andy','The Matrix'),(
                             #('Claude','Monty Python and the Holy Grail'),('Claude','Snakes on 
                             #('Frida','The Matrix'),('Frida','The Shawshank Redemption'),('Frid
                             #('Georgia','Anaconda'),('Georgia','Monty Python and the Holy Grail
                             #('Joan','Forest Gump'),('Joan','Kung Fu Panda'),('Joan','Mean Girl
                             #('Lee','Forest Gump'),('Lee','Kung Fu Panda'),('Lee','Mean Girls')
                             #('Pablo','The Dark Knight'),('Pablo','The Matrix'),('Pablo','The S
                             #('Vincent','The Godfather'),('Vincent','The Shawshank Redemption')
            B.add_edges_from(all_edges)
            return B# Your Answer Here
        plot_graph(answer_one())
```

1.0.2 Question 2

Using the graph from the previous question, add nodes attributes named `'type'` where movies have the value `'movie'` and employees have the value `'employee'` and return that graph.

This function should return a networkx graph with node attributes `{'type': 'movie'}` or `{'type': 'employee'}`

```
In [ ]: def answer_two():

            # Your Code Here
            B = answer_one()
            B.add_nodes_from(employees,bipartite=0, type = 'employee')
            B.add_nodes_from(movies,bipartite=1, type = 'movie')

            return B # Your Answer Here
        plot_graph(answer_two())
```

1.0.3 Question 3

Find a weighted projection of the graph from `answer_two` which tells us how many movies different pairs of employees have in common.

This function should return a weighted projected graph.

```
In [ ]: def answer_three():

            # Your Code Here
            B = answer_two()
            P = bipartite.weighted_projected_graph(B,employees)
            return P# Your Answer Here
        plot_graph(answer_three())
```

1.0.4 Question 4

Suppose you'd like to find out if people that have a high relationship score also like the same types of movies.

Find the Pearson correlation (using `DataFrame.corr()`) between employee relationship scores and the number of movies they have in common. If two employees have no movies in common it should be treated as a 0, not a missing value, and should be included in the correlation calculation.

This function should return a float.

```
In [155]: def answer_four():

              # Your Code Here
              df = pd.read_csv('Employee_Relationships.txt', delimiter = "\t",header=None)
              df.rename(columns={0:'employee1',1:'employee2',2:'score'},inplace=True)
              #df['movies_in_common'] = None
              #df_2 = pd.read_csv('Employee_Movie_Choices.txt', delimiter = "\t")
              df_1 = nx.to_pandas_dataframe(answer_three())
              df['shared_movies'] = df_1.lookup(df['employee1'], df['employee2'])
              #df_3 = df_2.groupby('#Employee')['Movie'].apply(list)
              #df_3 = df_3.reset_index()
              #my_list = []
              #for x in df_3["Movie"]:
                  #for y in df_3["Movie"]:
                      #if x != y:
                          #my_list.append(len(set(x) & set(y)))
              #final = pd.merge(df,df_3,on='employee1')
              corr_P = df['score'].corr(df['shared_movies'],method='pearson')
              return corr_P#final#.agg({"review_scores_value":np.average})# Your Answer Here
          answer_four()

Out[155]: 0.7883962221733474

In [ ]:
```