Graph Features

October 13, 2020

1 Creating a feature matrix from a networkx graph

In this notebook we will look at a few ways to quickly create a feature matrix from a networkx graph.

```
In [1]: import networkx as nx
        import pandas as pd

        G = nx.read_gpickle('major_us_cities')
```

1.1 Node based features

```
In [2]: G.nodes(data=True)
```

```
Out[2]: [('El Paso, TX', {'location': (-106, 31), 'population': 674433}),
         ('Long Beach, CA', {'location': (-118, 33), 'population': 469428}),
         ('Dallas, TX', {'location': (-96, 32), 'population': 1257676}),
         ('Oakland, CA', {'location': (-122, 37), 'population': 406253}),
         ('Albuquerque, NM', {'location': (-106, 35), 'population': 556495}),
         ('Baltimore, MD', {'location': (-76, 39), 'population': 622104}),
         ('Raleigh, NC', {'location': (-78, 35), 'population': 431746}),
         ('Mesa, AZ', {'location': (-111, 33), 'population': 457587}),
         ('Arlington, TX', {'location': (-97, 32), 'population': 379577}),
         ('Sacramento, CA', {'location': (-121, 38), 'population': 479686}),
         ('Wichita, KS', {'location': (-97, 37), 'population': 386552}),
         ('Tucson, AZ', {'location': (-110, 32), 'population': 526116}),
         ('Cleveland, OH', {'location': (-81, 41), 'population': 390113}),
         ('Louisville/Jefferson County, KY',
          {'location': (-85, 38), 'population': 609893}),
         ('San Jose, CA', {'location': (-121, 37), 'population': 998537}),
         ('Oklahoma City, OK', {'location': (-97, 35), 'population': 610613}),
         ('Atlanta, GA', {'location': (-84, 33), 'population': 447841}),
         ('New Orleans, LA', {'location': (-90, 29), 'population': 378715}),
         ('Miami, FL', {'location': (-80, 25), 'population': 417650}),
         ('Fresno, CA', {'location': (-119, 36), 'population': 509924}),
         ('Philadelphia, PA', {'location': (-75, 39), 'population': 1553165}),
         ('Houston, TX', {'location': (-95, 29), 'population': 2195914}),
```

```
('Boston, MA', {'location': (-71, 42), 'population': 645966}),
('Kansas City, MO', {'location': (-94, 39), 'population': 467007}),
('San Diego, CA', {'location': (-117, 32), 'population': 1355896}),
('Chicago, IL', {'location': (-87, 41), 'population': 2718782}),
('Charlotte, NC', {'location': (-80, 35), 'population': 792862}),
('Washington D.C.', {'location': (-77, 38), 'population': 646449}),
('San Antonio, TX', {'location': (-98, 29), 'population': 1409019}),
('Phoenix, AZ', {'location': (-112, 33), 'population': 1513367}),
('San Francisco, CA', {'location': (-122, 37), 'population': 837442}),
('Memphis, TN', {'location': (-90, 35), 'population': 653450}),
('Los Angeles, CA', {'location': (-118, 34), 'population': 3884307}),
('New York, NY', {'location': (-74, 40), 'population': 8405837}),
('Denver, CO', {'location': (-104, 39), 'population': 649495}),
('Omaha, NE', {'location': (-95, 41), 'population': 434353}),
('Seattle, WA', {'location': (-122, 47), 'population': 652405}),
('Portland, OR', {'location': (-122, 45), 'population': 609456}),
('Tulsa, OK', {'location': (-95, 36), 'population': 398121}),
('Austin, TX', {'location': (-97, 30), 'population': 885400}),
('Minneapolis, MN', {'location': (-93, 44), 'population': 400070}),
('Colorado Springs, CO', {'location': (-104, 38), 'population': 439886}),
('Fort Worth, TX', {'location': (-97, 32), 'population': 792727}),
('Indianapolis, IN', {'location': (-86, 39), 'population': 843393}),
('Las Vegas, NV', {'location': (-115, 36), 'population': 603488}),
('Detroit, MI', {'location': (-83, 42), 'population': 688701}),
('Nashville-Davidson, TN', {'location': (-86, 36), 'population': 634464}),
('Milwaukee, WI', {'location': (-87, 43), 'population': 599164}),
('Columbus, OH', {'location': (-82, 39), 'population': 822553}),
('Virginia Beach, VA', {'location': (-75, 36), 'population': 448479}),
('Jacksonville, FL', {'location': (-81, 30), 'population': 842583})]
```

```
In [3]: # Initialize the dataframe, using the nodes as the index
        df = pd.DataFrame(index=G.nodes())
```

1.1.1 Extracting attributes

Using `nx.get_node_attributes` it's easy to extract the node attributes in the graph into DataFrame columns.

```
In [ ]: df['location'] = pd.Series(nx.get_node_attributes(G, 'location'))
        df['population'] = pd.Series(nx.get_node_attributes(G, 'population'))

        df.head()
```

1.1.2 Creating node based features

Most of the networkx functions related to nodes return a dictionary, which can also easily be added to our dataframe.

```
In [ ]: df['clustering'] = pd.Series(nx.clustering(G))
        df['degree'] = pd.Series(G.degree())

        df
```

2 Edge based features

```
In [ ]: G.edges(data=True)
```

```
In [ ]: # Initialize the dataframe, using the edges as the index
        df = pd.DataFrame(index=G.edges())
```

2.0.1 Extracting attributes

Using `nx.get_edge_attributes`, it's easy to extract the edge attributes in the graph into DataFrame columns.

```
In [ ]: df['weight'] = pd.Series(nx.get_edge_attributes(G, 'weight'))

        df
```

2.0.2 Creating edge based features

Many of the networkx functions related to edges return a nested data structures. We can extract the relevant data using list comprehension.

```
In [ ]: df['preferential attachment'] = [i[2] for i in nx.preferential_attachment(G, df.index)]

        df
```

In the case where the function expects two nodes to be passed in, we can map the index to a lamda function.

```
In [ ]: df['Common Neighbors'] = df.index.map(lambda city: len(list(nx.common_neighbors(G, city[

        df
```