homework_1

September 24, 2024

1 Homework 1

1.1 Setup

1.1.1 Installing and Importing Packages

[1]:
```
!pip install networkx
```

```
Requirement already satisfied: networkx in
/Users/othmane123/opt/anaconda3/lib/python3.9/site-packages (2.7.1)
```

[2]:
```
import networkx as nx
```

1.1.2 Creating the graphs

Create two NetworkX graph objects for Facebook and Twitter. The graph for Facebook is undirected but the graph for Twitter is Directed.

[3]:
```
facebook = nx.Graph()
twitter = nx.DiGraph()
```

Read the edges and construct the graph for Facebook - It is important to note that facebook's graph is undirected and that The 'ego' node does not appear, but it is assumed that they follow every node id that appears in the file.

[4]:
```
f = open("Facebook-Ego/3437_2.edges", "r")
fb_edges = []

for line in f:
  node_1 = int(line.split(" ")[0])
  node_2 = int(line.split(" ")[1])
  fb_edges.append((node_1, node_2))
f.close()

facebook.add_edges_from(fb_edges)
```

Read the edges and construct the graph for Twitter - It is important to note that Edges are directed (a follows b). The 'ego' node does not appear, but it is assumed that they follow every node id that appears in this file.

[5]:
```
f = open("Twitter-Ego/6408382.edges", "r")
twitter_edges = []

for line in f:
  node_1 = int(line.split(" ")[0])
  node_2 = int(line.split(" ")[1])
  twitter_edges.append((node_1, node_2))
f.close()

twitter.add_edges_from(twitter_edges)
```

1.2 (a) How many nodes and edges are there in the networks?

[6]:
```
fb_nodes_count = facebook.number_of_nodes()
fb_edges_count = facebook.number_of_edges()
print(f"There are {fb_nodes_count} nodes and {fb_edges_count} edges in the␣
 ↪facebook graph")
```

```
There are 132 nodes and 367 edges in the facebook graph
```

[7]:
```
twitter_nodes_count = twitter.number_of_nodes()
twitter_edges_count = twitter.number_of_edges()
print(f"There are {twitter_nodes_count} nodes and {twitter_edges_count} edges␣
 ↪in the twitter graph")
```

```
There are 151 nodes and 3379 edges in the twitter graph
```

1.3 (b) What are the maximum degree and the average degree of the networks?

1.3.1 Facebook

[8]:
```
degrees = dict(facebook.degree())
max_degree = max(degrees.values())

print("The maximum degree of the facebook network is:", max_degree)
```

```
The maximum degree of the facebook network is: 25
```

[9]:
```
total_degrees = sum(dict(facebook.degree()).values())
num_nodes = facebook.number_of_nodes()
average_degree = total_degrees / num_nodes

print("The average degree of the facebook network is:", average_degree)
```

```
The average degree of the facebook network is: 5.5606060606060606
```

1.3.2 Twitter

[10]:
```
degrees = dict(twitter.degree())

max_degree = max(degrees.values())

print("The maximum degree of the twitter network is:", max_degree)
```

```
The maximum degree of the twitter network is: 139
```

[11]:
```
total_degrees = sum(dict(twitter.degree()).values())
num_nodes = twitter.number_of_nodes()
average_degree = total_degrees / num_nodes

print("The average degree of the twitter network is:", average_degree)
```

```
The average degree of the twitter network is: 44.75496688741722
```

1.4 (c) Extract 5 - 8 nodes from the network and state them as a partial network. What is the adjacency matrix of the partial network? Why do we need adjacency matrix to describe the structure of the network?

1.4.1 Facebook

We start first by extracting some nodes from the graph

[12]:
```
nodes_to_extract = [3710,3713,3633,3596,3702,3627,3718,3584]
partial_network = facebook.subgraph(nodes_to_extract)
```

Then we get the adjacency matrix of the partial network

[13]:
```
matrix = nx.to_numpy_array(partial_network, nodelist=nodes_to_extract)
matrix
```

[13]:
```
array([[0., 1., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 1., 0., 1., 1., 0., 0.],
       [0., 0., 0., 1., 0., 1., 0., 0.],
       [0., 0., 0., 1., 1., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 0., 0., 1., 0.]])
```

1.4.2 Twitter

We start first by extracting some nodes from the graph

[14]:
```
nodes_to_extract = [390660853, 79024978, 64838619, 16434310, 93598339,␣
↪37878528, 16475403, 15039741]
partial_network = twitter.subgraph(nodes_to_extract)
```

Then we get the adjacency matrix of the partial network

[15]:
```
matrix = nx.to_numpy_array(partial_network, nodelist=nodes_to_extract)
matrix
```

[15]:
```
array([[0., 1., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0.],
       [0., 1., 0., 1., 0., 1., 1., 0.],
       [0., 1., 1., 0., 1., 1., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 1.],
       [0., 0., 0., 1., 0., 0., 1., 0.]])
```

1.4.3 Why do we need adjacency matrix to describe the structure of the network?

Adjacency matrices provide a compact and clear representation of networks, and while they might be memory consuming in some cases they are great for use cases like these, where we have a small/partial network. The adjacency matrix representation can be used for all cases wether the graph is directed or undirectd, weighted or unweighted. Matrices are also ideal for mathematical analyses using linear algebra but also for various computer algorithms for shortest path, clustering etc.