Applied social network analysis in Python

Week 1:

Network is a set of nodes with interconnections(edges)

1. Python representation:
   - Import networkx as nx
   - G = nx.Graph() # Graph / G = nx.DiGraph()
   - G.add_edge('a', 'b') create and edge
2. Some edges carry more weight than others
3. Some network carry information about friendship and antoganist based on conflict or disagreement
4. Edges can carry relation
5. Multigraps: A network where multiple edges can connect the same nodes (parallel graphs, nx.MultiGraphs())
6. G.edges() – get list of all edges

G.edges(data=True) – list of all edges with attributes

7. G.edge['A']['B'] – dictrionarry of attributes of edge A and B
8. G.add_node('Name', role='Name of node's role')
9. G.nodes() – list of nodes

G.nodes(data=True) – list of all nodes with attributes

10. Bipartite graphs: a graph whose nodes can be split into two sets L and R and every egde connects and node L and node in R
11. From network.algorithms import bipartite:

B = nx.Graph()

B.add_nodes_from([list of nodes], bipartite=0) label one set of nodes 0

B.add_nodes_from([list of nodes], bipartite=1) label other set of nodes 1

B.add_edges_from([('A', 1)…) create edges

12. Bipartite.projected_graph(B, X)

Week 2:

13. Tradic closure: The tendency for people who share connections in a social network to become connected
14. Local Clustering Coefficient:
    - Find the degree of and Node
    - Find pairs of nodes ((degree*(degree-1))/2)
    - Find pairs of nodes that are connected and divide by pairs of nodes
15. nx.clustering(graph, 'node')
16. Clustering coeff measures the degree to wich nodes in a network tend to cluster or form triangles
17. Local Clustering Coeff fraction of pairs the nodes friends that are friends with each other
18. Global Clustering Coeff:
    - Average local clustering – nx.average_clustering()
    - Transitivity – ration of triangles and numbers of 'open triads' nx.transitivity

19. Breadth-first search – a systematic and efficient procedure for computing distance from node to all other nodes in a large network by discovering nodes in layers(nx.bfs_tree(prag, 'node')
20. Average distance between pairs of nodes nx.average_shortest_path_lenght
21. Maximum distance between any pairs of nodes nx.diameter()
22. Eccentricity of node is the largest distance between n and all other nodes nx.eccentricity()
23. Radius of a graph is the minimum eccentricity nx.radius()
24. Periphery is a set of nodes that have eccentricity equal to a diameter
25. The center of graph is the set of nodes that have eccentricity equal to radius nx.center()
26. An undirected graph is connected if for every pair nodes there is a path between them
27. Nx.node_connected_component(graph, 'node') returns nodes that belongs to node
28. A directed graph is strongly connected of for every pair nodes, there is a directed path from one node to other and vice versa nx.is_strongly_connected
29. Network robustness the ability of a network to maintain its general structural properties when its faces failers or attacks
30. Nx.node_connectivity() returns the number of nodes that are connected
31. Nx.minimum_node_cut() return the node by deleting which graph become disconnected
32. Nx.edge_connectivity() returns the number of edges that are connected
33. Nx.minimum_edge_cut() return edges by deleting which graph become disconnected
34. Nx.weakly_connected_components() return nodes weakly connected components

Week 3:

35. Important nodes have many connections nx.degree_centrality()/ nx.in_degree_centrality()/nx.out_degree_centrality()
36. Closness centrality – important nodes are close to other nodes nx.closeness_centrality()
37. In graph theory, betweenness centrality is a measure of centrality in a graph based on shortest paths. For every pair of vertices in a connected graph, there exists at least one shortest path between the vertices such that either the number of edges that the path passes through (for unweighted graphs) or the sum of the weights of the edges (for weighted graphs) is minimized. The betweenness centrality for each vertex is the number of these shortest paths that pass through the vertex.
38. Nx.betwenness_centrality(graph, normalized=, endpoints=true/false)
39. Normalization pf betweenness centrality – devide by numbers of pairs of node
40. Approximation – approximate computation by taking subsets of nodes
41. Subsets – we can define subsets of source and target nodes to compute betweenness centrality
42. Edge betweenness centrality – we can apply the same framework to find important edges instead of nodes
43. Basic PageRank(nx.pagerank(graph, alpha=)):
    - All Nodes start with PageRank of 1/n(num of nodes)

- Perform the basic PageRank Update rule - each node gives an equal share of its current pagerank to all the nodes it links to. The new pagerank of each node is the sum of all pagerank it received from other nodes

44. Hubs and Authorities, given a query to a search engine:
    - Root: set of higly relevant web pages – potential authorities
    - Find all pages that link to a page in root – potential hubs
    - Base: root nodes and any node that links to a node in a root
45. Hits algorithm hx.hits(graph):
    - Assign each node an authority and hub score of 1
    - Apply the Authority update rule – each node's authority score is the sum of hub scores of each node that points to it
    - Apply the hub update rule – each node's hub score is the sum of authority scores of each node that it points to
    - Normalize authority score
    - Repeat k time

Week 4:

46. Degree Distribution – is the probability distribution of the degrees over the entire network
47. Nx.Barbasi_albert_graph(n, m) – returns a network with n nodes. Each new node attaches to m existing nodes according to the preferential attachment model
48. Small World Model – start with a ring of n nodes, where each node is connected to its k nearest neighbors\
49. Nx.watts_strogatz_graph(n, k, p) – return a small world network with n nodes, starting with a ring lattice with each node connected to its k nearest neighbors an rewriting probability p
50. Link prediction problem – Given a network, predict which edges will be formed in the future
51. Basic measures for link prediction:
    - Number of common neighbors nx.common_neighbors(Graph)
    - Jaccard Coefficient – number of common neighbors normalized by the total number of neighbors nx.jaccard_coefficient(Graph)
    - Resource Allocation Index – fraction of resource that a node can send to another through their common neighbors nx.ressource_allocation_index(G)
    - Adam-Adar index – similar to resource allocation index but with log in the denominator nx.adamic_adar_index(G)
    - Preferential attachments – nodes with high degree get more neighbors nx.preferential_attachment(g)
    - Common Neibor Soundarajah-Hopcroft Score – Number of common neighbors but with bonus for neighbors in same community nx.cn_soundarajan_hopcroft(G)