Homework 2

专业：大数据001

学号：2184114639

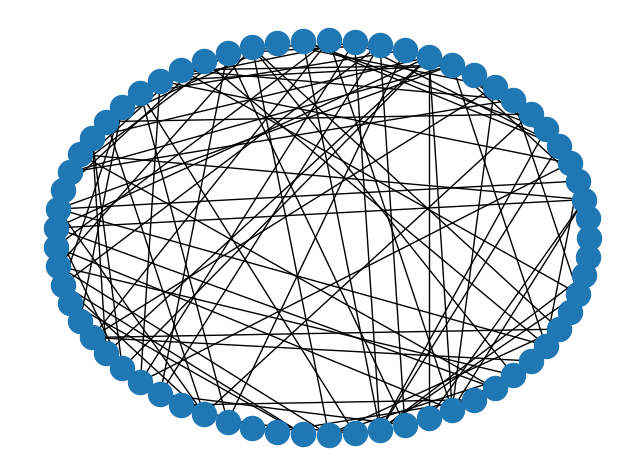
姓名：郅啸淇

日期：2023年5月1日星期一

1. 利用课堂上提供的一个社交网络数据，寻找在各种指标测度下最重要的三个节点。提交代码和结果。

1.1 你是用的社交网络是什么？请提供可视化图。

第一个课件中anodes和aedges



可视化图

代码

1. **import** networkx as nx
2. **import** matplotlib.pyplot as plt
3. **import** pandas as pd
5. data\_nodes = pd.read\_csv("anodes.csv")
6. data\_edges  = pd.read\_csv("aedges.csv")
8. nodes = data\_nodes["Id"]
9. source = data\_edges["Source"]
10. target = data\_edges["Target"]
12. G = nx.Graph()
13. G.add\_nodes\_from(nodes)
14. edges = []
15. **for** i **in** range(len(source)):
16. edges.append((source[i],target[i]))
17. G.add\_edges\_from(edges)
18. pos = nx.circular\_layout(G)
19. **print**(nx.adjacency\_matrix(G).todense())
20. **print**(len(nodes),len(edges))
21. nx.draw(G,pos)
22. plt.show()

1.2 提供各种指标测度下最重要的三个节点，以及相关测度。可以考虑使用表格的形式展示结果。

测度：度D，度中心性DC，介数中心性BC，接近中心性CC，K核KC，特征向量中心性EC，PageRank。

运行结果

1. D        DC        BC        CC  KC        EC        PR
2. 0  8  0.125000  0.334747  0.212312   3  0.425510  0.031093
3. 1  7  0.109375  0.322917  0.204106   3  0.321590  0.031039
4. 2  6  0.093750  0.320933  0.202153   3  0.307633  0.024822

代码

1. **import** networkx as nx
2. **import** matplotlib.pyplot as plt
3. **import** pandas as pd
5. data\_nodes = pd.read\_csv("anodes.csv")
6. data\_edges = pd.read\_csv("aedges.csv")
8. nodes = data\_nodes["Id"]
9. source = data\_edges["Source"]
10. target = data\_edges["Target"]
12. G = nx.Graph()
13. G.add\_nodes\_from(nodes)
14. edges = []
15. **for** i **in** range(len(source)):
16. edges.append((source[i], target[i]))
17. G.add\_edges\_from(edges)
19. # 度和度中心性
20. degree = dict(G.degree())
21. degree\_centrality = dict(nx.degree\_centrality(G))
23. new\_degree = sorted(degree.values(), reverse=True)
24. new\_degree\_centrality = sorted(degree\_centrality.values(), reverse=True)
26. # print(new\_degree[0:3])
27. # print(new\_degree\_centrality[0:3])
28. new\_degree = pd.DataFrame(new\_degree[0:3])
29. new\_degree\_centrality = pd.DataFrame(new\_degree\_centrality[0:3])
31. # 介数中心性
33. bc = nx.betweenness\_centrality(G)
34. new\_bc = sorted(bc.values(), reverse=True)
35. # print(new\_bc[0:3])
36. new\_bc = pd.DataFrame(new\_bc[0:3])
38. # 接近中心性
40. cc = nx.closeness\_centrality(G)
41. new\_cc = sorted(cc.values(), reverse=True)
42. # print(new\_cc[0:3])
43. new\_cc = pd.DataFrame(new\_cc[0:3])
45. # k核
46. core = nx.core\_number(G)
47. new\_core = sorted(core.values(), reverse=True)
48. # print(new\_core[0:3])
49. new\_core = pd.DataFrame(new\_core[0:3])
51. # 特征向量中心性
52. ec = nx.eigenvector\_centrality(G, max\_iter=1000)
53. new\_ec = sorted(ec.values(), reverse=True)
54. # print(new\_ec[0:3])
55. new\_ec = pd.DataFrame(new\_ec[0:3])
57. # PageRank
58. pr = nx.pagerank(G)
59. new\_pr = sorted(pr.values(), reverse=True)
60. # print(new\_pr[0:3])
61. new\_pr = pd.DataFrame(new\_pr[0:3])
63. df1 = pd.concat(
64. [new\_degree[0:3], new\_degree\_centrality[0:3], new\_bc[0:3], new\_cc[0:3], new\_core[0:3], new\_ec[0:3], new\_pr[0:3]],
65. axis=1)
66. new\_name = ['D','DC','BC','CC','KC','EC','PR']
67. df1.columns = new\_name
68. **print**(df1)

2. 基于节点相似性度量，利用AA指标，构建好友推荐函数，为上述社交网络中的节点进行好友推荐。

2.1 请提供好友推荐函数代码。并适当标注。

1. preds = nx.adamic\_adar\_index(G)
2. data = []# data为adamic\_adar\_index函数返回值序列
3. # u为自己，v为目标，p为AA值
4. **for** u, v, p **in** preds:
5. # 如果v已经是u的好友，则不加入函数返回值序列
6. **if** v **in** G.neighbors(u):
7. **continue**
8. # 如果v是u本身，则不加入函数返回值序列
9. **if** v == u:
10. **continue**
11. data.append([u,v,p])
12. recommendation = [] # 此为最终结果，包含自己，最佳推荐好友和AA值
13. AA = 0 # AA存储着每一个u的当前最大值
14. # nx.adamic\_adar\_index返回的迭代有一定规律，会连续返回u对各个v的AA值，故u的变化是连续的（例如：1 1 1 1 2 2 2 2 3 3 3）
15. # 故采用u是否等于下一位u的值来确定该序列是否结束，若结束，则将此时的u，最大的潜在好友v和最大值返回到最终结果序列中
16. **for** i **in** range(len(data)):
17. u = data[i][0]
18. v = data[i][1]
19. p = data[i][2]
20. **if** i == len(data) - 1:# 如果达到最后一位，则直接判断返回
21. **if** p > AA:# 如果当前p值大于最大值，则替代
22. AA = p
23. v\_temp = v # 记录更新最大值时候对应的v，即潜在推荐好友
24. recommendation.append([u,v\_temp,AA])
25. **else**:
26. u\_next = data[i+1][0] # 判断u是否变化
27. **if** u == u\_next: # 若无变化，尝试更新最大值
28. **if** p > AA:
29. AA = p
30. v\_temp = v # 记录更新最大值时候对应的v，即潜在推荐好友
31. **else**: # 若有变化，则该u序列结束，返回此时u和最大的潜在好友v
32. recommendation.append([u,v\_temp,AA])
33. AA = 0 # 结束一个u序列后，重置最大值
34. df = pd.DataFrame(recommendation,columns=['Node','Best\_Target','AA\_Value'])
35. **print**(recommendation)
36. **print**(df)

2.2 请提供为每个节点推荐的排名第一的好友列表。

第一列为节点Id，第二列为第一推荐好友Id

1. Node                       Best\_Target  \
2. 0   3CB0D0299D1CADAF47D61818F6545F0E  3CB0D0299D1CADAFDD6105CF4BD36EB6
3. 1   8C8C8DAC8F75B1825A740CE3933C35B8  3CB0D0299D1CADAF99C012ED0CD7F6E9
4. 2   3CB0D0299D1CADAF6F9E1F26EFC672F4  3CB0D0299D1CADAF99C012ED0CD7F6E9
5. 3   3CB0D0299D1CADAF9BEDCB6976161BE3  3CB0D0299D1CADAF976144CD0472EB6F
6. 4   3CB0D0299D1CADAFDD6105CF4BD36EB6  3CB0D0299D1CADAF9AF1A1FAF3549B0C
7. 5   3CB0D0299D1CADAF22AD3A36F54AF92B  3CB0D0299D1CADAF6CD52C094641043D
8. 6   3CB0D0299D1CADAF0AD3568DCCFD8AC3  3CB0D0299D1CADAF6CD52C094641043D
9. 7   3CB0D0299D1CADAFD54F419BD14B4BE0  3CB0D0299D1CADAF6CD52C094641043D
10. 8   3CB0D0299D1CADAFC2389DCC24D6640B  3CB0D0299D1CADAFA7A673506759EE9C
11. 9   3CB0D0299D1CADAF2D6B0E6AF904261E  3CB0D0299D1CADAF5C6D69738E47563C
12. 10  3CB0D0299D1CADAF1C01DE4AC7503791  3CB0D0299D1CADAF5C6D69738E47563C
13. 11  3CB0D0299D1CADAF4510C43458469B6A  3CB0D0299D1CADAFE1241EADD04404F6
14. 12  3CB0D0299D1CADAFBA01C68BE8CA166C  3CB0D0299D1CADAFA6B575375CC4D3AB
15. 13  3CB0D0299D1CADAF67CA8E6EE34B4D61  9A05CFD0426F3F70328E748150865666
16. 14  3CB0D0299D1CADAFB618D3DEA2D8EDF0  3CB0D0299D1CADAF40887359FC19595B
17. 15  3CB0D0299D1CADAFAABBE19BB6565C5B  3CB0D0299D1CADAF9A464F7BA49C453A
18. 16  E948FC6883590A0526B542072F3B5C60  3CB0D0299D1CADAF9A464F7BA49C453A
19. 17  3CB0D0299D1CADAF5364E3E6C926E827  3CB0D0299D1CADAF40887359FC19595B
20. 18  3CB0D0299D1CADAF9A464F7BA49C453A  3CB0D0299D1CADAFFF24B32A1BF779B9
21. 19  3CB0D0299D1CADAFDE9F70BFB5C5C76F  3CB0D0299D1CADAFAC9503ECB2C6DFA6
22. 20  3CB0D0299D1CADAF4F0DF52F0BD2B5D5  3CB0D0299D1CADAF364847EF03B226B2
23. 21  3CB0D0299D1CADAF40887359FC19595B  3CB0D0299D1CADAF4EFFF900D36AB841
24. 22  3CB0D0299D1CADAFFF24B32A1BF779B9  3CB0D0299D1CADAF750E0434CB7AF64E
25. 23  3CB0D0299D1CADAFF0C931CF8CF2B0B5  3CB0D0299D1CADAF1E8F4DFBCC019789
26. 24  3CB0D0299D1CADAF960799F3044141EB  3CB0D0299D1CADAF1E8F4DFBCC019789
27. 25  3CB0D0299D1CADAF976144CD0472EB6F  3CB0D0299D1CADAFE426D2A4D0A48D37
28. 26  9A05CFD0426F3F70328E748150865666  3CB0D0299D1CADAF2E5434795146D728
29. 27  3CB0D0299D1CADAF8F37944C7EAB960B  3CB0D0299D1CADAF364847EF03B226B2
30. 28  3CB0D0299D1CADAFB260E73DA5517D73  3CB0D0299D1CADAF1E8F4DFBCC019789
31. 29  3CB0D0299D1CADAFE426D2A4D0A48D37  3CB0D0299D1CADAFA252FE0F154619D3
32. 30  3CB0D0299D1CADAF223FC9601C16B9DB  3CB0D0299D1CADAF65FCE1EE83327F20
33. 31  3CB0D0299D1CADAF1E8F4DFBCC019789  3CB0D0299D1CADAF446E2D5BA3CD59C8
34. 32  3CB0D0299D1CADAF2A7017C242AF0399  3CB0D0299D1CADAF4EFFF900D36AB841
35. 33  3CB0D0299D1CADAF4EFFF900D36AB841  3CB0D0299D1CADAF227414263FD70CDA
36. 34  3CB0D0299D1CADAF61CFD06FFA0D3EFB  3CB0D0299D1CADAF2E5434795146D728
37. 35  3CB0D0299D1CADAF65FCE1EE83327F20  3CB0D0299D1CADAF9AF1A1FAF3549B0C
38. 36  3CB0D0299D1CADAFCBF6F2E21EFAAB76  3CB0D0299D1CADAF9AF1A1FAF3549B0C
39. 37  3CB0D0299D1CADAFE1241EADD04404F6  3CB0D0299D1CADAF9AF1A1FAF3549B0C
40. 38  3CB0D0299D1CADAFE27E2AFEE4722CCC  3CB0D0299D1CADAFA6B575375CC4D3AB
41. 39  3CB0D0299D1CADAF2E5434795146D728  3CB0D0299D1CADAFE21085F94B121EFA
42. 40  3CB0D0299D1CADAFEDDCDB76FA1215A1  3CB0D0299D1CADAFE21085F94B121EFA
43. 41  3CB0D0299D1CADAF3666E4BEDC6F01A1  3CB0D0299D1CADAFE21085F94B121EFA
44. 42  3CB0D0299D1CADAF227414263FD70CDA  3CB0D0299D1CADAFE21085F94B121EFA
45. 43  3CB0D0299D1CADAFA2B0F5D31B22D3E8  3CB0D0299D1CADAFFFC1582BBDB3D74D
46. 44  3CB0D0299D1CADAFFA289A673F4A7FD7  3CB0D0299D1CADAFAC9503ECB2C6DFA6
47. 45  3CB0D0299D1CADAF5CFD8BA17D746632  3CB0D0299D1CADAFAC9503ECB2C6DFA6
48. 46  3CB0D0299D1CADAFDBDA505781D87372  3CB0D0299D1CADAFAC9503ECB2C6DFA6
49. 47  3CB0D0299D1CADAF750E0434CB7AF64E  3CB0D0299D1CADAFA7A673506759EE9C
50. 48  3CB0D0299D1CADAFE21085F94B121EFA  3CB0D0299D1CADAF6CD52C094641043D
51. 49  3CB0D0299D1CADAFA252FE0F154619D3  3CB0D0299D1CADAFAC9503ECB2C6DFA6
52. 50  3CB0D0299D1CADAF6CD52C094641043D  3CB0D0299D1CADAF9AF1A1FAF3549B0C
53. 51  3CB0D0299D1CADAF9AF1A1FAF3549B0C  3CB0D0299D1CADAFA6B575375CC4D3AB
54. 52  3CB0D0299D1CADAFA6B575375CC4D3AB  3CB0D0299D1CADAFA6B575375CC4D3AB
55. 53  3CB0D0299D1CADAF99C012ED0CD7F6E9  3CB0D0299D1CADAFA6B575375CC4D3AB
56. 54  3CB0D0299D1CADAFBF4ECA897F1741EC  3CB0D0299D1CADAFA6B575375CC4D3AB
57. 55  3CB0D0299D1CADAFFFC1582BBDB3D74D  3CB0D0299D1CADAF364847EF03B226B2
58. 56  3CB0D0299D1CADAFEB47CA10657AEA97  3CB0D0299D1CADAFAC9503ECB2C6DFA6
59. 57  3CB0D0299D1CADAF364847EF03B226B2  3CB0D0299D1CADAFFE37ADDE91EDA35F
60. 58  3CB0D0299D1CADAFFE37ADDE91EDA35F  3CB0D0299D1CADAF446E2D5BA3CD59C8
61. 59  3CB0D0299D1CADAFAC9503ECB2C6DFA6  3CB0D0299D1CADAF5C6D69738E47563C
62. 60  3CB0D0299D1CADAF446E2D5BA3CD59C8  3CB0D0299D1CADAF5C6D69738E47563C
63. 61  3CB0D0299D1CADAF5C6D69738E47563C  3CB0D0299D1CADAF5C6D69738E47563C
64. 62  3CB0D0299D1CADAFA7A673506759EE9C  3CB0D0299D1CADAF5C6D69738E47563C
65. 63  3CB0D0299D1CADAFAEF88023F619D9FA  3CB0D0299D1CADAF5C6D69738E47563C

3. 自己设计一个节点相似性指标，并用python代码实现。寻找一个社交网络，验证自己的节点相似性指标。

3.1 请说明设计的节点相似性指标的思路，原理，以及其优点。

ZXQ指标：

思路：

分子：x和y的共同好友数

分母：x好友数加y好友数再取对数

ZXQ指标越大，则节点相似度越高

首先共同好友数越多代表两节点相似度越高，但同时考虑到如果两节点都拥有非常多的好友，则共同好友的影响会被削弱，这种指标考虑到了两个节点本身的状况。

3.2 请提供Python代码。

1. **import** networkx as nx
2. **import** matplotlib.pyplot as plt
3. **import** pandas as pd
4. **import** math
6. pd.set\_option('display.max\_columns',None)
7. pd.set\_option('display.max\_rows',None)
9. # 导入数据
10. data\_nodes = pd.read\_csv("anodes.csv")
11. data\_edges = pd.read\_csv("aedges.csv")
13. # 提取节点和链接
14. nodes = data\_nodes["Id"]
15. source = data\_edges["Source"]
16. target = data\_edges["Target"]
18. # 生成图并插入节点和链接
19. G = nx.Graph()
20. G.add\_nodes\_from(nodes)
21. edges = []
22. **for** i **in** range(len(source)):
23. edges.append((source[i], target[i]))
24. G.add\_edges\_from(edges)
25. result = [] # 结果存储
27. **def** ZXQindex(u,v): # 指标计算函数
28. k\_u = len(list(G.neighbors(u))) # u好友数
29. k\_v = len(list(G.neighbors(v))) # v好友数
30. n = len(list(set(G.neighbors(u)) & set(G.neighbors(v)))) # 共同好友数
31. **if** (k\_u + k\_v) <= 1:# 排除分母为零的情况
32. result.append([u,v,None])
33. **return**
34. ZXQ = n / math.log(k\_u + k\_v)
35. result.append([u,v,ZXQ])
37. **for** u **in** nodes:
38. **for** v **in** nodes:
39. **if** v == u: # 若v为u本身，不再计算
40. **continue**
41. ZXQindex(u,v)
42. # for i in result:
43. #     print(i[2])
44. **print**(result)

3.2 请说明你使用的验证网络。举例说明你设计的相似性指标的科学性。

验证网络节点为学生，边代表学生之间的联系。在较为狭窄的校园生活中，两个人共同好友数较大程度受到二者自身好友数的影响，仅凭共同好友数判断相似度准确度不高，故引入两人好友数进行平衡。