به نام خدا

درس : شبکه های اجتماعی

پروژه یک

استاد: دکتر مریم حسینی

اعضای گروه:
زهرا حیدری

فاطمه سادات رضوی

سوال 1:

رسم گراف با استفاده از کتابخانه های panda , networkx :

برای رسم گراف داده ها ابتدا داده ها یعنی گره ها و گره هایی که توسل یالها با هم در ارتباط هستند در یک فایل متنی ذخیره می کنیم و با استفاده از Pd.read_cvs () فایل داده ای دریافت می شود.

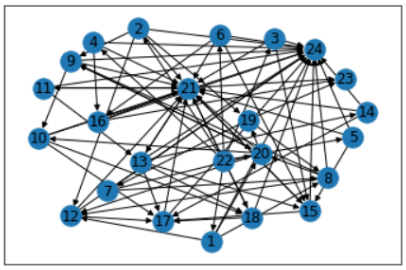
بااستفاده از تابع from-pandas-edgelist() گراف مربوط به فایل خوانده شده ذخیره میشود.

با استفاده از Digraph گراف را جهت دار در نظر میگیریم وسپس با تابع Draw گراف رسم میشود.

Data set 1:

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

22]:
```
df=pd.read_csv('arcs.txt', sep=' ', names=['n1','n2','Weight'])
G=nx.from_pandas_edgelist(df,'n1','n2',create_using=nx.DiGraph)
k_pos=nx.spring_layout(G, k=2.0)
nx.draw_networkx(G,k_pos)
```



درشکل خروجی نمایش گراف که متشکل از گره هایی که توسط یالها باهم ارتباط دارند.

با تابع layout گره های متصل شده توسط یالها نزدیک به هم قرار دارند.

همچنین پارامتر k برای فاصله میان گره هاست.

تعداد یالها و گره ها ی مجموعه داده اول:

با استفاده از تابع edges میتوان گره هایی که با یال به هم متصل هستند با استفاده از تابع nodes() گره های گراف و توابع number-of-edges() و number-of-nodes تعداد یالها و گره ها در خروجی نمایش داده می شوند.

```
[23]: G.edges()

[23]: OutEdgeView([(22, 1), (22, 2), (22, 3), (22, 4), (22, 5), (22, 6), (1, 24), (1, 8), (1, 12), (1, 20), (2, 24), (2, 9), (2, 12), (2, 19), (2, 21), (3, 24), (3, 15), (3, 21), (4, 24), (4, 16), (4, 21), (5, 24), (5, 15), (5, 21), (6, 24), (6, 15), (6, 16), (6, 21), (16, 23), (16, 24), (16, 21), (19, 23), (19, 7), (19, 8), (21, 23), (21, 11), (21, 13), (21, 15), (21, 18), (7, 24), (7, 8), (7, 20), (8, 24), (8, 9), (8, 12), (8, 20), (9, 24), (9, 10), (9, 20), (9, 21), (10, 24), (10, 17), (10, 18), (10, 21), (11, 24), (11, 13), (11, 21), (12, 24), (12, 21), (12, 17), (13, 24), (13, 14), (13, 15), (13, 17), (13, 18), (13, 21), (14, 24), (14, 17), (14, 21), (15, 24), (15, 17), (15, 21), (17, 24), (17, 21), (18, 24), (18, 17), (18, 21), (20, 9), (20, 10), (20, 12), (20, 19), (20, 21)])

[24]: G.nodes()

[24]: NodeView((22, 1, 2, 3, 4, 5, 6, 16, 23, 19, 21, 24, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 20))

[25]: G.number_of_edges()

[25]: 82

[26]: G.number_of_nodes()

[26]: 24
```

مجموعه داده 2

```
#data set twitter
DFA=pd.read_csv('twitter_combined.txt', sep=' ', names=['n1','n2'])
DFA
```

	n1	n2
0	214328887	34428380
1	17116707	28465635
2	380580781	18996905
3	221036078	153460275
4	107830991	17868918
...	...	...
2420761	99841247	154263215
2420762	99841247	194403468
2420763	99841247	180165101
2420764	99841247	253509115
2420765	99841247	463410501

2420766 rows × 2 columns

```
G2=nx.from_pandas_edgelist(DFA,'n1','n2',create_using=nx.DiGraph)
#nx.draw_networkx(G2)
```

تعداد یال و گره های مجموعه داده دوم:

```
G2.number_of_edges()
```

```
1768149
```

```
G2.number_of_nodes()
```

```
81306
```

سوال دوم :

ابتدا گراف مجموعه داده را با روش گفته شده در سوال یک می سازیم

سپس با توابع مجود درکتابخانه های سوال یک قطر، شعاع ، درجه مرکزیت ، بینابینی ، برداروِیژه و pagerank را درخروجی مشاهده می کنیم:

قطر با استفاده از تابع nx.diameter(G):

شعاع با تابع nx.radius(G)

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

2]: df=pd.read_csv('arcs.txt', sep=' ', names=['n1','n2','Weight'])
    G=nx.from_pandas_edgelist(df,'n1','n2',create_using=nx.Graph)

3]: nx.diameter(G)

3]: 3

4]: nx.radius(G)

4]: 2
```

مقدار مرکزیت درجه:

تابعdegree_centrality(Graph_name)مرکزیت گراف را درخروجی نمایش میدهد

بر اساس این مرکزیت،در یک گراف غیرجهتدار،هر چه یک گره دارای درجه بالاتری باشد، دارای مرکزیت درجه بالاتری است.درجه یک گره،

تعداد یالهای متصل به آن است.

```
nx.degree_centrality(G)
```

```
{22: 0.2608695652173913,
 1: 0.21739130434782608,
 2: 0.2608695652173913,
 3: 0.17391304347826086,
 4: 0.17391304347826086,
 5: 0.17391304347826086,
 6: 0.21739130434782608,
 16: 0.21739130434782608,
 23: 0.13043478260869565,
 19: 0.21739130434782608,
 21: 0.7391304347826086,
 24: 0.7826086956521738,
 7: 0.17391304347826086,
 8: 0.30434782608695654,
 9: 0.2608695652173913,
 10: 0.2608695652173913,
 11: 0.13043478260869565,
 12: 0.30434782608695654,
 13: 0.30434782608695654,
 14: 0.17391304347826086,
 15: 0.30434782608695654,
 17: 0.34782608695652173,
 18: 0.21739130434782608,
 20: 0.34782608695652173}
```

مرکزیت نزدیکی:

در کتابخانه Networkx از تابع closeness_centrality برای محاسبه مرکزیت درجه استفاده میشود.

در این تابع ارامتر اول، گراف موردنظر است، پارامتر دوم گرهای است که میخواهیم مقدار مرکزیت نزدیکی فقط برای آن

محاسبه شود، پارامتر سوم یک ویژگی از یالهامثلا وزن است که اگر نوشته شود، کوتاهترین مسیر بر اساس آنها محاسبه میشود.

```
n [6]: nx.closeness_centrality(G)

ut[6]: {22: 0.4791666666666667,
 1: 0.5476190476190477,
 2: 0.575,
 3: 0.5348837209302325,
 4: 0.5348837209302325,
 5: 0.5348837209302325,
 6: 0.5476190476190477,
 16: 0.5609756097560976,
 23: 0.5111111111111111,
 19: 0.45098039215686275,
 21: 0.7931034482758621,
 24: 0.8214285714285714,
 7: 0.5348837209302325,
 8: 0.5897435897435898,
 9: 0.575,
 10: 0.5609756097560976,
 11: 0.5111111111111111,
 12: 0.5897435897435898,
 13: 0.5609756097560976,
 14: 0.5227272727272727,
 15: 0.575,
 17: 0.575,
 18: 0.5348837209302325,
 20: 0.6052631578947368}
```

بینابینی:

با استفاده ازتابع()betweenss-centrality :

مقدار این مرکزیت بر اساس موقعیت گرهها در شبکه و قرار گرفتنآنها در کوتاهترین مسیر میان جفت گرههای دیگر محاسبه می شود.

```
nx.betweenness_centrality(G)
```

```
{22: 0.023197816676077544,
 1: 0.014978927528334643,
 2: 0.037832649344507044,
 3: 0.008315668740569925,
 4: 0.00893913947273631,
 5: 0.008315668740569925,
 6: 0.012103547528448715,
 16: 0.011923583662714097,
 23: 0.01197063805759458,
 19: 0.018324549846288975,
 21: 0.26407882302151076,
 24: 0.30752511651721137,
 7: 0.00768868812347073,
 8: 0.019985569985569978,
 9: 0.007155163676902806,
 10: 0.007811029550159985,
 11: 0.00026350461133069827,
 12: 0.01656785243741765,
 13: 0.00849802371541502,
 14: 0.00026350461133069827,
 15: 0.012721155655938263,
 17: 0.013109354413702236,
 18: 0.0012516469038208167,
 20: 0.03883845622976058}
```

بردارویژه و Page rank

In [8]:
```
nx.eigenvector_centrality(G)
```

Out[8]:
```
{22: 0.11879320033164695,
 1: 0.14563318558533997,
 2: 0.18556822631955108,
 3: 0.1468309367906904,
 4: 0.13758790616760433,
 5: 0.1468309367906904,
 6: 0.1663094057737224,
 16: 0.15228909261647527,
 23: 0.08212315879986454,
 19: 0.0974026107207723,
 21: 0.39237262642197596,
 24: 0.4122526167205757,
 7: 0.11379569477884902,
 8: 0.1790739029765314,
 9: 0.20248128968815782,
 10: 0.21283509230819364,
 11: 0.13208315439452714,
 12: 0.22752540868722496,
 13: 0.2280453042132963,
 14: 0.16572254472034778,
 15: 0.22455497095138957,
 17: 0.26300427080324673,
 18: 0.19294521993013458,
 20: 0.20095505235999375}
```

[9]:
```
nx.pagerank(G)
```

[9]:
```
{22: 0.040959250571295895,
 1: 0.033347393229418675,
 2: 0.03904779977381961,
 3: 0.0277346066766039,
 4: 0.028229020807731577,
 5: 0.0277346066766039,
 6: 0.03365585159813139,
 16: 0.0348314767943331,
 23: 0.023312777473857522,
 19: 0.03525077848752567,
 21: 0.10298220997367312,
 24: 0.1081388877401876,
 7: 0.02821635378877245,
 8: 0.045146932687636754,
 9: 0.038302878452865036,
 10: 0.03810415245590017,
 11: 0.021934666691122394,
 12: 0.04382229749029902,
 13: 0.044710864282869106,
 14: 0.027183039411740564,
 15: 0.044692336797281906,
 17: 0.049396153149038134,
 18: 0.032581158121369316,
 20: 0.05068450686792298}
```

[ ]: