Exercício 5 T1-PE

February 4, 2021

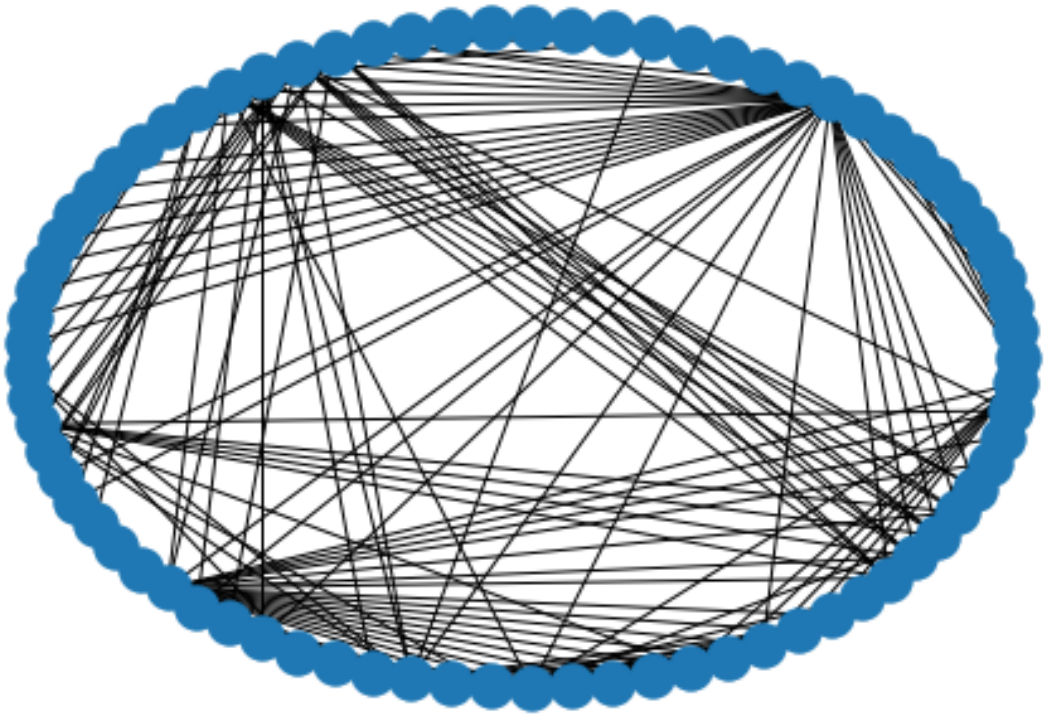
1 Exercício 5

Inicialmente, vamos definir o grafo que representa as conexões entre cada personagem do livro.

[142]:
```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import random as r

g= nx.read_gml("lesmis.gml") # Read the network
g = g.to_undirected() #Remove a direcao dos links

npos=nx.circular_layout(g)
nx.draw(g,pos = npos, with_labels=False, arrows=False)
plt.show()
```



1.0.1 Implementação PageRank

Primeiramente vamos definir e mostrar a respectiva matriz de transição do grafo.

[144]:
```
T = nx.to_numpy_matrix(g)
print(T)
```

```
[[0. 1. 1. … 0. 0. 0.]
 [1. 0. 0. … 0. 0. 0.]
 [1. 0. 0. … 0. 0. 0.]
 …
 [0. 0. 0. … 0. 0. 0.]
 [0. 0. 0. … 0. 0. 0.]
 [0. 0. 0. … 0. 0. 0.]]
```

Agora vamos aplicar o ajuste estocástico na matriz T e já transformar ela uma matriz de probabilidade de transição, a aplicação do ajuste estocástico é necessário pois para o algoritmo pagerank funcionar é necessário uma cadeia ergódica.

[9]:
```
N = T.shape[0]
P = np.zeros((N,N))
for i in range(0,N):
    for j in range(0,N):
        if(np.sum(T[i,:]) > 0):
            P[i,j] = T[i,j]/np.sum(T[i,:])
        else:
            P[i,j] = 1/N
print(P)
```

```
[[0.         0.1        0.1        … 0.         0.         0.        ]
 [1.         0.         0.         … 0.         0.         0.        ]
 [0.33333333 0.         0.         … 0.         0.         0.        ]
 …
 [0.         0.         0.         … 0.         0.         0.        ]
 [0.         0.         0.         … 0.         0.         0.        ]
 [0.         0.         0.         … 0.         0.         0.        ]]
```

Agora podemos fazer o ajuste ergódico para obter a matriz G (ou matriz de Google).

[10]:
```
G = np.zeros((N,N))
alpha = 0.85
for i in range(0,N):
    for j in range(0,N):
        G[i,j] = alpha*P[i,j] + (1-alpha)/N
print(G)
```

```
[[0.00194805 0.08694805 0.08694805 … 0.00194805 0.00194805 0.00194805]
 [0.85194805 0.00194805 0.00194805 … 0.00194805 0.00194805 0.00194805]
 [0.28528139 0.00194805 0.00194805 … 0.00194805 0.00194805 0.00194805]
```

```
…
[0.00194805 0.00194805 0.00194805 … 0.00194805 0.00194805 0.00194805]
[0.00194805 0.00194805 0.00194805 … 0.00194805 0.00194805 0.00194805]
[0.00194805 0.00194805 0.00194805 … 0.00194805 0.00194805 0.00194805]]
```

Finalmente com a matriz G, para obter PageRank, é necessário obter a matriz estacionária de G, ou simplesmente obter os autovetores referente ao autovalor 1.

[124]:
```
eigvals, eigvecs = np.linalg.eig(G.T)
eigvec1 = eigvecs[:,np.isclose(eigvals, 1)]
eigvec1 = eigvec1[:,0]

stationary = eigvec1 / eigvec1.sum()

stationary = stationary.real
```

Com a classificação de cada nó do PageRank em mãos, vamos fazer várias caminhadas aleatórias, começando uma de cada nó, dando 1000 passos aleatórios e vamos ir somando as frequências de visitas em cada vetor, para poder comparar com o PageRank. Após a contagem das frequências vamos dividir pelo valor total de visitas em todos os nós para chegar em uma distribuição estacionária estimada.

1.0.2 Implementação Caminhada Aleatória

Para implementação das caminhadas aleatórias, primeiramente vamos definir um dicionário contendo todos os nomes dos personagens, cada inicialmente com o valor 0, que será somado em 1 cada vez que a caminhada passar pelo seu respectivo nó na caminhada aleatória. Depois pegamos todos esses valores e dividimos pela soma do total das frequência, assim obtemos valores entre 0 e 1 para poder comparar com o PageRank.

[151]:
```
freq = {}
for i in list(g):
    temp = {i: 0}
    freq.update(temp)

steps = 1000

for i in list(g):
    atual = i
    for j in range(1,steps):
        atual= r.choice(list(g.neighbors(atual)))
        freq[atual] = freq[atual] + 1

freq_est = np.array(list(freq.values()))
freq_est = freq_est/sum(freq_est)
```

1.0.3 Comparação

Agora que temos os scores do algoritimo PageRank e os valores das frequências referente as caminhadas aleatórias, vamos mostrar para cada nome de personagem o seu valor referente em cada método.

[152]:
```
for i in range(0,N):
    print(list(g)[i], ': PG:', stationary[i], 'RW:', freq_est[i])
```

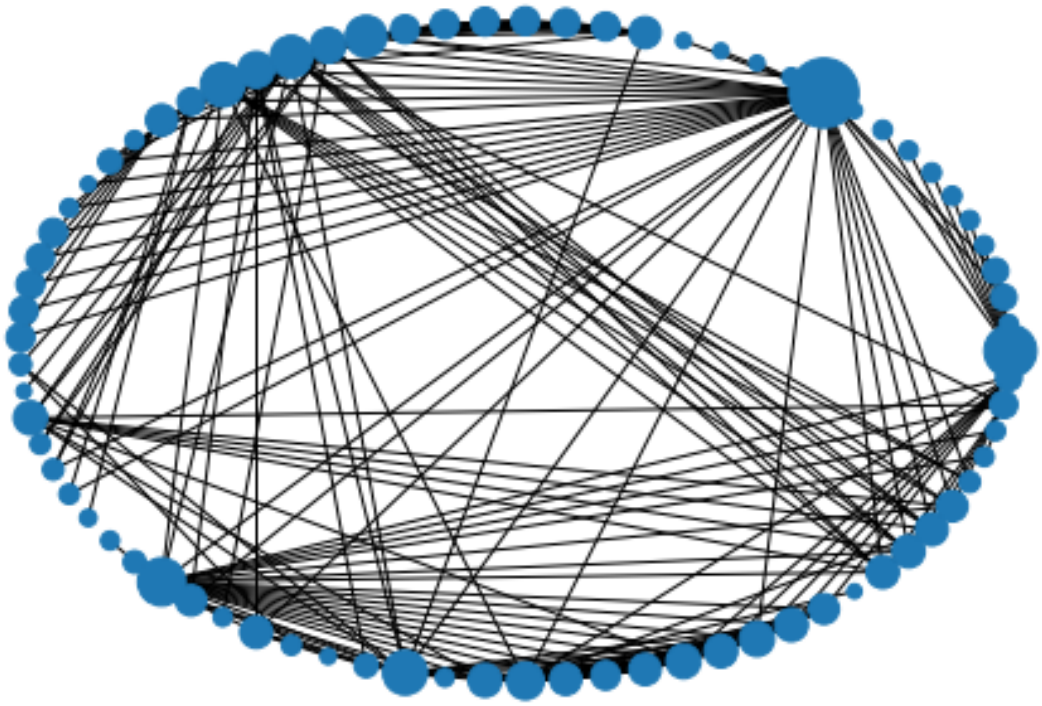
```
Myriel : PG: 0.04277928102271197 RW: 0.02154102154102154
Napoleon : PG: 0.005584290834982474 RW: 0.00208000208000208
MlleBaptistine : PG: 0.010277134629737873 RW: 0.005668005668005668
MmeMagloire : PG: 0.010277134629737816 RW: 0.005876005876005876
CountessDeLo : PG: 0.005584290834982454 RW: 0.002444002444002444
Geborand : PG: 0.005584290834982457 RW: 0.002457002457002457
Champtercier : PG: 0.005584290834982466 RW: 0.002054002054002054
Cravatte : PG: 0.005584290834982466 RW: 0.002262002262002262
Count : PG: 0.005584290834982466 RW: 0.001885001885001885
OldMan : PG: 0.005584290834982466 RW: 0.002379002379002379
Labarre : PG: 0.0037290409310482635 RW: 0.002145002145002145
Valjean : PG: 0.07543012163278477 RW: 0.07107107107107107
Marguerite : PG: 0.005260327543023245 RW: 0.003991003991003991
MmeDeR : PG: 0.0037290409310482635 RW: 0.001768001768001768
Isabeau : PG: 0.0037290409310482635 RW: 0.001807001807001807
Gervais : PG: 0.0037290409310482635 RW: 0.002197002197002197
Tholomyes : PG: 0.0156474273684825 RW: 0.018304018304018305
Listolier : PG: 0.012618202914107783 RW: 0.014118014118014117
Fameuil : PG: 0.012618202914107783 RW: 0.014235014235014234
Blacheville : PG: 0.012618202914107783 RW: 0.013572013572013573
Favourite : PG: 0.012618202914107783 RW: 0.013936013936013935
Dahlia : PG: 0.012618202914107783 RW: 0.013585013585013584
Zephine : PG: 0.012618202914107783 RW: 0.013897013897013897
Fantine : PG: 0.02702270491720569 RW: 0.03075803075803076
MmeThenardier : PG: 0.019501134691061076 RW: 0.022347022347022346
Thenardier : PG: 0.027926525694033526 RW: 0.032188032188032185
Cosette : PG: 0.020611215084857627 RW: 0.0210990210990211
Javert : PG: 0.03030273590581364 RW: 0.0349050349050349
Fauchelevent : PG: 0.011638047873644034 RW: 0.00796900796900797
Bamatabois : PG: 0.015576264721022286 RW: 0.01638001638001638
Perpetue : PG: 0.005407488540755646 RW: 0.003848003848003848
Simplice : PG: 0.009073646968135077 RW: 0.008333008333008334
Scaufflaire : PG: 0.0037290409310482635 RW: 0.001898001898001898
Woman1 : PG: 0.005244177726338937 RW: 0.003848003848003848
Judge : PG: 0.012424659363823528 RW: 0.011661011661011661
Champmathieu : PG: 0.012424659363823528 RW: 0.011414011414011414
Brevet : PG: 0.012424659363823528 RW: 0.011622011622011623
Chenildieu : PG: 0.012424659363823528 RW: 0.011843011843011843
Cochepaille : PG: 0.012424659363823528 RW: 0.011713011713011713
Pontmercy : PG: 0.0073680972010412545 RW: 0.006045006045006045
```

```
Boulatruelle : PG: 0.0034316486255474878 RW: 0.001963001963001963
Eponine : PG: 0.017793911863244048 RW: 0.02200902200902201
Anzelma : PG: 0.0063135385865619835 RW: 0.006344006344006344
Woman2 : PG: 0.0068368625283506575 RW: 0.005629005629005629
MotherInnocent : PG: 0.006202126104197613 RW: 0.004004004004004004
Gribier : PG: 0.004421137121201304 RW: 0.001768001768001768
Jondrette : PG: 0.005265424107402563 RW: 0.001742001742001742
MmeBurgon : PG: 0.00780558155141321 RW: 0.003783003783003783
Gavroche : PG: 0.035767318194729704 RW: 0.04344604344604345
Gillenormand : PG: 0.014957475581397065 RW: 0.013702013702013701
Magnon : PG: 0.005271222702569829 RW: 0.003874003874003874
MlleGillenormand : PG: 0.01626020862966739 RW: 0.012805012805012806
MmePontmercy : PG: 0.006010133393378015 RW: 0.004121004121004121
MlleVaubois : PG: 0.003922505853082997 RW: 0.001716001716001716
LtGillenormand : PG: 0.008713597430429626 RW: 0.007501007501007501
Marius : PG: 0.030894936215123014 RW: 0.03647803647803648
BaronessT : PG: 0.005146458723386866 RW: 0.00416000416000416
Mabeuf : PG: 0.017478022290751023 RW: 0.0205010205010205
Enjolras : PG: 0.021882033328144528 RW: 0.029575029575029575
Combeferre : PG: 0.01589212469810006 RW: 0.02122902122902123
Prouvaire : PG: 0.013145880776009436 RW: 0.016926016926016925
Feuilly : PG: 0.01589212469810006 RW: 0.020267020267020267
Courfeyrac : PG: 0.01857844249302665 RW: 0.024466024466024465
Bahorel : PG: 0.01719987801838601 RW: 0.022334022334022333
Bossuet : PG: 0.01895953011022147 RW: 0.0247000247000247
Joly : PG: 0.01719987801838601 RW: 0.0228020228020228
Grantaire : PG: 0.014456669473094019 RW: 0.01948701948701949
MotherPlutarch : PG: 0.0032986263977917967 RW: 0.001716001716001716
Gueulemer : PG: 0.016691838036280715 RW: 0.01929201929201929
Babet : PG: 0.016691838036280715 RW: 0.020761020761020762
Claquesous : PG: 0.016561020318792915 RW: 0.019916019916019916
Montparnasse : PG: 0.01517092849358998 RW: 0.016757016757016758
Toussaint : PG: 0.0068368625283506575 RW: 0.006201006201006201
Child1 : PG: 0.005791254017601826 RW: 0.004498004498004498
Child2 : PG: 0.005791254017601826 RW: 0.004511004511004511
Brujon : PG: 0.011866660942688598 RW: 0.014248014248014248
MmeHucheloup : PG: 0.010689825140848529 RW: 0.013624013624013625
```

Agora com os valores PageRank e os valores resultantes da caminhada aleatória para cada nó, vamos plotar o grafo novamente, com o tamanho de cada nó influenciado pelo seu respectivo score, começando pelo PageRank.

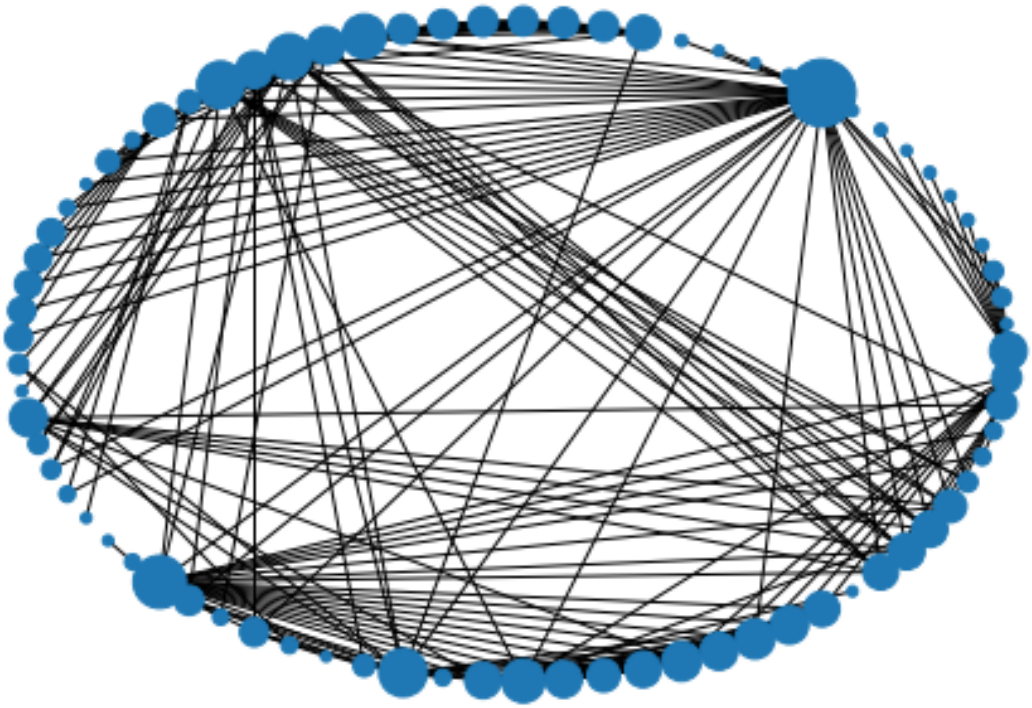
[153]:
```
npos=nx.circular_layout(g,scale=1)

nx.draw(g,pos = npos, with_labels=False, node_size = stationary*10000,␣
 ↪arrows=False)
plt.draw()
plt.show()
```



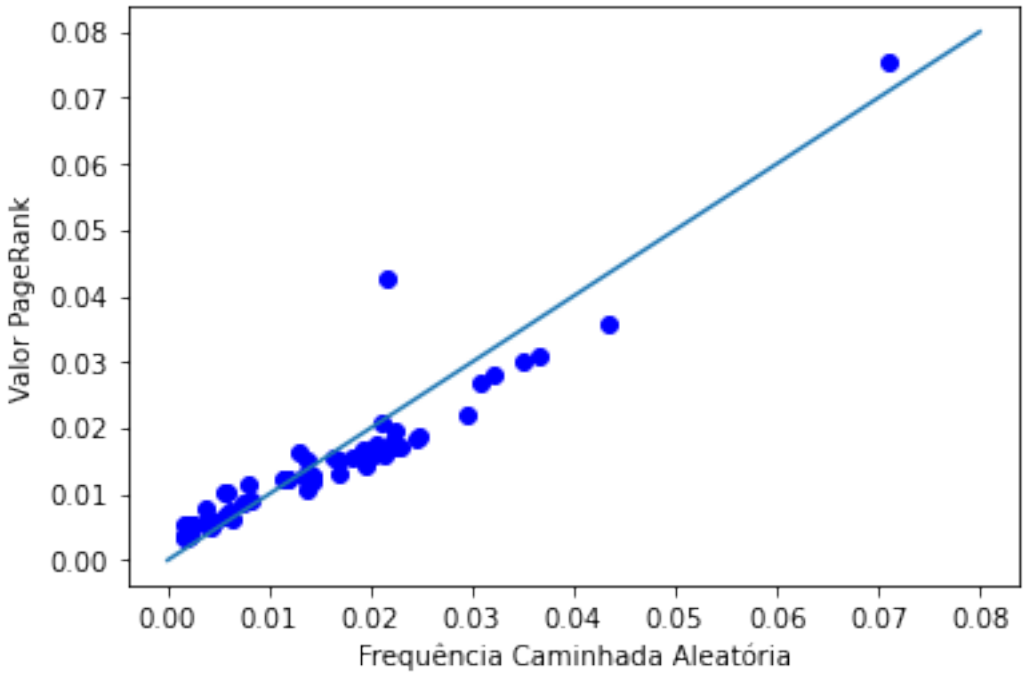
Agora caminhada aleatória.

[154]:
```
nx.draw(g,pos = npos, with_labels=False, node_size = freq_est*10000,␣
↪arrows=False)
plt.draw()
plt.show()
```



Como podemos perceber, os grafos ficaram parecidos, com o tamanho dos nós quase iguais, então para confirmar, agora vamos comparar os valores do PageRank e os valores da caminhada aleatória fazendo um gráfico de dispersão.

[156]:
```
plt.plot(freq_est,stationary,'bo')
plt.plot([0,0.08],[0,0.08])
plt.xlabel('Frequência Caminhada Aleatória')
plt.ylabel('Valor PageRank')
plt.show()
```



Como podemos ver os valores são bem parecidos e possuem correlação bem forte.