

```
import networkx as nx
from networkx.algorithms import bipartite
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import numpy as np
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
# you need to have tensorflow
from stellargraph.data import UniformRandomMetaPathWalk
from stellargraph import StellarGraph
```

```
!pip install networkx==2.3
```

```
Requirement already satisfied: networkx==2.3 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (2.3)
Requirement already satisfied: decorator>=4.3.0 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from networkx==2.3)
(5.1.1)
```

```
!pip install stellargraph
```

```
Requirement already satisfied: stellargraph in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (1.2.1)
Requirement already satisfied: networkx>=2.2 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from stellargraph)
(2.3)
Requirement already satisfied: scikit-learn>=0.20 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from stellargraph)
(1.0.2)
Requirement already satisfied: numpy>=1.14 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from stellargraph)
(1.21.6)
Requirement already satisfied: gensim>=3.4.0 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from stellargraph)
(4.2.0)
Requirement already satisfied: tensorflow>=2.1.0 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from stellargraph)
(2.10.0)
Requirement already satisfied: pandas>=0.24 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from stellargraph)
(1.3.5)
Requirement already satisfied: matplotlib>=2.2 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from stellargraph)
(2.2.3)
Requirement already satisfied: scipy>=1.1.0 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from stellargraph)
(1.7.3)
Requirement already satisfied: smart-open>=1.8.1 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from gensim>=3.4.0-
>stellargraph) (6.1.0)
```

Requirement already satisfied: Cython==0.29.28 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from gensim>=3.4.0->stellargraph) (0.29.28)

Requirement already satisfied: cycler>=0.10 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from matplotlib>=2.2->stellargraph) (0.11.0)

Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from matplotlib>=2.2->stellargraph) (1.4.4)

Requirement already satisfied: pytz in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from matplotlib>=2.2->stellargraph) (2022.2.1)

Requirement already satisfied: python-dateutil>=2.1 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from matplotlib>=2.2->stellargraph) (2.8.2)

Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,! =2.1.6,>=2.0.1 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from matplotlib>=2.2->stellargraph) (3.0.9)

Requirement already satisfied: six>=1.10 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from matplotlib>=2.2->stellargraph) (1.16.0)

Requirement already satisfied: decorator>=4.3.0 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from networkx>=2.2->stellargraph) (5.1.1)

Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from scikit-learn>=0.20->stellargraph) (3.1.0)

Requirement already satisfied: joblib>=0.11 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from scikit-learn>=0.20->stellargraph) (1.1.0)

Requirement already satisfied: wrapt>=1.11.0 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (1.14.1)

Requirement already satisfied: absl-py>=1.0.0 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (1.2.0)

Requirement already satisfied: keras-preprocessing>=1.1.1 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (1.1.2)

Requirement already satisfied: astunparse>=1.6.0 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (1.6.3)

Requirement already satisfied: h5py>=2.9.0 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (3.7.0)

Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (4.3.0)

Requirement already satisfied: termcolor>=1.1.0 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from

tensorflow>=2.1.0->stellargraph) (1.1.0)
Requirement already satisfied: tensorboard<2.11,>=2.10 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (2.10.0)
Requirement already satisfied: flatbuffers>=2.0 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (2.0.7)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (0.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (3.3.0)
Requirement already satisfied: tensorflow-estimator<2.11,>=2.10.0 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (2.10.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (1.48.1)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (3.19.4)
Requirement already satisfied: packaging in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (21.3)
Requirement already satisfied: keras<2.11,>=2.10.0 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (2.10.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (14.0.6)
Requirement already satisfied: setuptools in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (63.4.1)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (0.2.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorflow>=2.1.0->stellargraph) (0.27.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from astunparse>=1.6.0->tensorflow>=2.1.0->stellargraph) (0.37.1)
Requirement already satisfied: markdown>=2.6.8 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow>=2.1.0->stellargraph) (3.4.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow>=2.1.0->stellargraph) (0.4.6)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\

amitk\anaconda3\envs\graphclustering\lib\site-packages (from
tensorboard<2.11,>=2.10->tensorflow>=2.1.0->stellargraph) (2.11.0)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from
tensorboard<2.11,>=2.10->tensorflow>=2.1.0->stellargraph) (2.2.2)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in c:\
users\amitk\anaconda3\envs\graphclustering\lib\site-packages (from
tensorboard<2.11,>=2.10->tensorflow>=2.1.0->stellargraph) (1.8.1)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from
tensorboard<2.11,>=2.10->tensorflow>=2.1.0->stellargraph) (2.28.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0
in c:\users\amitk\anaconda3\envs\graphclustering\lib\site-packages
(from tensorboard<2.11,>=2.10->tensorflow>=2.1.0->stellargraph)
(0.6.1)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.11,>=2.10->tensorflow>=2.1.0-
>stellargraph) (4.9)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\
amitk\anaconda3\envs\graphclustering\lib\site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.11,>=2.10->tensorflow>=2.1.0-
>stellargraph) (0.2.8)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\
amitk\anaconda3\envs\graphclustering\lib\site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.11,>=2.10->tensorflow>=2.1.0-
>stellargraph) (5.2.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\
amitk\anaconda3\envs\graphclustering\lib\site-packages (from google-
auth-oauthlib<0.5,>=0.4.1->tensorboard<2.11,>=2.10->tensorflow>=2.1.0-
>stellargraph) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in c:\users\
amitk\anaconda3\envs\graphclustering\lib\site-packages (from
markdown>=2.6.8->tensorboard<2.11,>=2.10->tensorflow>=2.1.0-
>stellargraph) (4.12.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\
amitk\anaconda3\envs\graphclustering\lib\site-packages (from
requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow>=2.1.0-
>stellargraph) (1.26.12)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\
amitk\anaconda3\envs\graphclustering\lib\site-packages (from
requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow>=2.1.0-
>stellargraph) (2.1.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from
requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow>=2.1.0-
>stellargraph) (3.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from
requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow>=2.1.0-

```

>stellargraph) (2022.6.15)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from
werkzeug>=1.0.1->tensorboard<2.11,>=2.10->tensorflow>=2.1.0-
>stellargraph) (2.1.1)
Requirement already satisfied: zipp>=0.5 in c:\users\amitk\anaconda3\
envs\graphclustering\lib\site-packages (from importlib-metadata>=4.4-
>markdown>=2.6.8->tensorboard<2.11,>=2.10->tensorflow>=2.1.0-
>stellargraph) (3.8.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from pyasn1-
modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10-
>tensorflow>=2.1.0->stellargraph) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\amitk\
anaconda3\envs\graphclustering\lib\site-packages (from requests-
oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1-
>tensorboard<2.11,>=2.10->tensorflow>=2.1.0->stellargraph) (3.2.1)

!pip install chardet

Requirement already satisfied: chardet in c:\users\amitk\anaconda3\
envs\graphclustering\lib\site-packages (5.0.0)

data=pd.read_csv('C:\\Users\\amitk\\OneDrive\\Desktop\\
movie_actor_network.csv', index_col=False, names=['movie','actor'])

edges = [tuple(x) for x in data.values.tolist()]

B = nx.Graph()
B.add_nodes_from(data['movie'].unique(), bipartite=0, label='movie')
B.add_nodes_from(data['actor'].unique(), bipartite=1, label='actor')
B.add_edges_from(edges, label='acted')

A = list(nx.connected_component_subgraphs(B))[0]

print("number of nodes", A.number_of_nodes())
print("number of edges", A.number_of_edges())

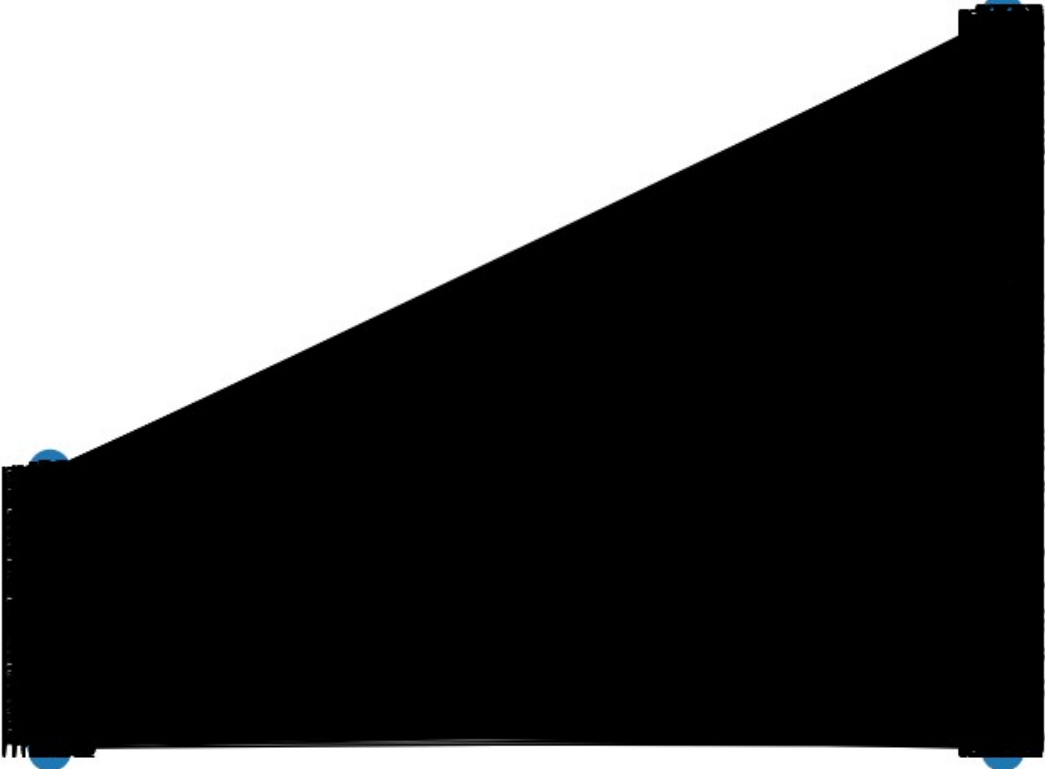
number of nodes 4703
number of edges 9650

l, r = nx.bipartite.sets(A)
pos = {}

pos.update((node, (1, index)) for index, node in enumerate(l))
pos.update((node, (2, index)) for index, node in enumerate(r))

nx.draw(A, pos=pos, with_labels=True)
plt.show()

```



```
movies = []
actors = []
for i in A.nodes():
    if 'm' in i:
        movies.append(i)
    if 'a' in i:
        actors.append(i)
print('number of movies ', len(movies))
print('number of actors ', len(actors))
```

```
number of movies  1292
number of actors  3411
```

```
movies = []
actors = []
for i in A.nodes():
    if 'm' in i:
        movies.append(i)
    if 'a' in i:
        actors.append(i)
print('number of movies ', len(movies))
print('number of actors ', len(actors))
```

```
number of movies  1292
number of actors  3411
```

```

# Create the random walker
rw = UniformRandomMetaPathWalk(StellarGraph(A))

# specify the metapath schemas as a list of lists of node types.
metapaths = [
    ["movie", "actor", "movie"],
    ["actor", "movie", "actor"]
]

walks = rw.run(nodes=list(A.nodes()), # root nodes
               length=100, # maximum length of a random walk
               n=1, # number of random walks per root node
               metapaths=metapaths
               )

print("Number of random walks: {}".format(len(walks)))

Number of random walks: 4703

from gensim.models import Word2Vec
model = Word2Vec(walks, vector_size=128, window=5)

model.wv.vectors.shape # 128-dimensional vector for each node in the
graph

(4703, 128)

# Retrieve node embeddings and corresponding subjects
node_ids = model.wv.index_to_key # list of node IDs
node_embeddings = model.wv.vectors # numpy.ndarray of size number of
nodes times embeddings dimensionality
node_targets = [ A.node[node_id]['label'] for node_id in node_ids]

def data_split(node_ids,node_targets,node_embeddings):
    '''In this function, we will split the node embeddings into
    actor_embeddings , movie_embeddings '''
    actor_nodes,movie_nodes=[],[]
    actor_embeddings,movie_embeddings=[],[]
    for i in range(len(node_ids)):
        if 'm' in node_targets[i]:
            movie_nodes.append(node_ids[i])
            movie_embeddings.append(node_embeddings[i])
        else:
            actor_nodes.append(node_ids[i])
            actor_embeddings.append(node_embeddings[i])

    return actor_nodes,movie_nodes,actor_embeddings,movie_embeddings

actor_nodes,movie_nodes,actor_embeddings,movie_embeddings=data_split(n
ode_ids,node_targets,node_embeddings)

```

```

def grader_actors(data):
    assert(len(data)==3411)
    return True
grader_actors(actor_nodes)

True

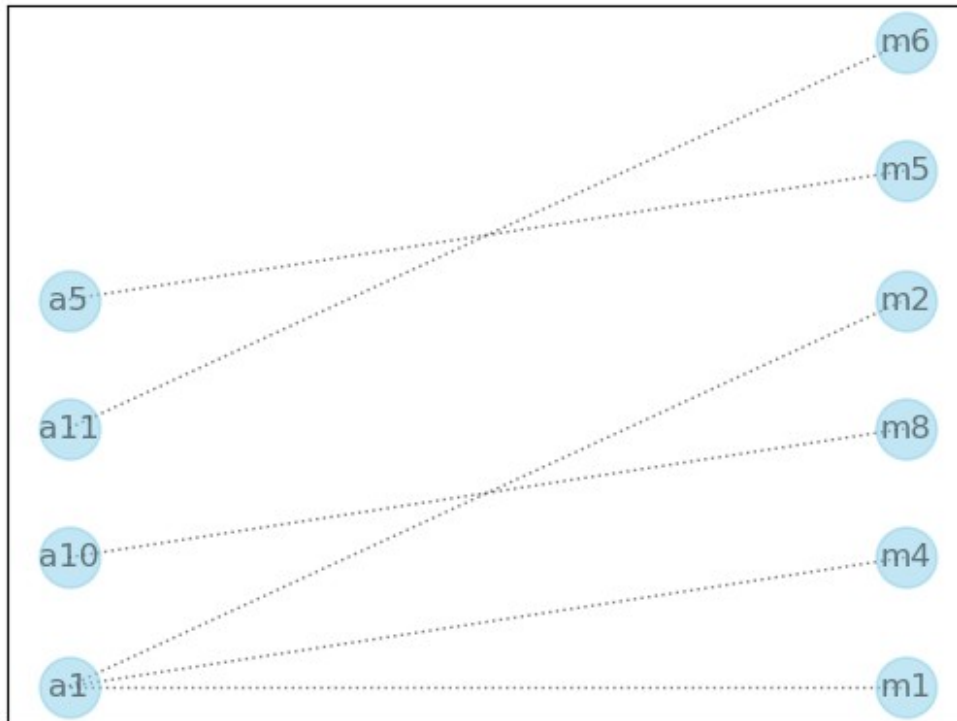
def grader_movies(data):
    assert(len(data)==1292)
    return True
grader_movies(movie_nodes)

True

def cost_1(graph,cluster):
    Gc = max(nx.connected_component_subgraphs(graph), key=len)
    connected=Gc.number_of_nodes()
    total_nodes=graph.number_of_nodes()
    value=((1/cluster)*(connected/total_nodes))
    return value


import networkx as nx
from networkx.algorithms import bipartite
graded_graph= nx.Graph()
graded_graph.add_nodes_from(['a1','a5','a10','a11'], bipartite=0) #
Add the node attribute "bipartite"
graded_graph.add_nodes_from(['m1','m2','m4','m6','m5','m8'],
bipartite=1)
graded_graph.add_edges_from([('a1','m1'),('a1','m2'),('a1','m4'),
('a11','m6'),('a5','m5'),('a10','m8')])
l={'a1','a5','a10','a11'};r={'m1','m2','m4','m6','m5','m8'}
pos = {}
pos.update((node, (1, index)) for index, node in enumerate(l))
pos.update((node, (2, index)) for index, node in enumerate(r))
nx.draw_networkx(graded_graph, pos=pos,
with_labels=True,node_color='skyblue',alpha=0.5,style='dotted',node_si
ze=500)

```

```

graded_cost1=cost_1(graded_graph,3)
def grader_cost1(data):
    assert(data==((1/3)*(4/10))) # 1/3 is number of clusters
    return True
grader_cost1(graded_cost1)

True

def cost_2(graph,cluster):
    degree=graph.degree()
    degree_value=0 # to store sum of all degree of actor nodes
    mov_nodes=0 #to store sum of all unique actor nodes
    for i in degree:
        if "a" in i[0]:
            degree_value=degree_value+i[1]
        else:
            mov_nodes=mov_nodes+1
    value=((1/cluster)*(degree_value/mov_nodes))
    return value

graded_cost2=cost_2(graded_graph,3)
def grader_cost2(data):
    assert(data==((1/3)*(6/6))) # 1/3 is number of clusters
    return True
grader_cost2(graded_cost2)

True

```

```

from sklearn.cluster import KMeans

cost_value={}
for clu in [3,5,10,50,200,300]:
    cost1=0
    cost2=0
    label=[]
    algo = KMeans(n_clusters=clu)
    algo.fit(actor_embeddings) # FITTING WITH KMEANS ALGO
    label=algo.labels_
    for i in range(clu):
        G1=nx.Graph() # drawing graph for each cluster
        label_divi=[]
        k=[index for index, value in enumerate(label) if value == i]#
        accessing each cluster based on labels_
        label_divi=[ actor_nodes[l] for l in k]
        for node in label_divi:
            sub_graph1=nx.ego_graph(B,node)
            G1.add_nodes_from(sub_graph1.nodes) # adding nodes
            G1.add_edges_from(sub_graph1.edges()) # adding edges
        cst1=cost_1(G1,clu)
        cost1=cost1+cst1 # calculating cost functions
        cst2=cost_2(G1,clu)
        cost2=cost2+cst2
        value=cost1*cost2
        cost_value[clu]=value

optimal_cluster_number = max(cost_value, key=cost_value.get)
print("optimal cluster number = ",optimal_cluster_number)
#https://www.geeksforgeeks.org/python-get-key-with-maximum-value-in-dictionary/

optimal_cluster_number = 3

model= KMeans(n_clusters=optimal_cluster_number)
model.fit(actor_embeddings)
label=model.labels_

node_cluster={}
for i in range(optimal_cluster_number):
    k=[index for index, value in enumerate(label) if value == i] #
    getting the cluster number for each node
    label_divi=[ actor_nodes[l] for l in k]
    for node in label_divi:
        node_cluster[node]=i

node_cluster['a1435']

0

from sklearn.manifold import TSNE
transform = TSNE #PCA

```

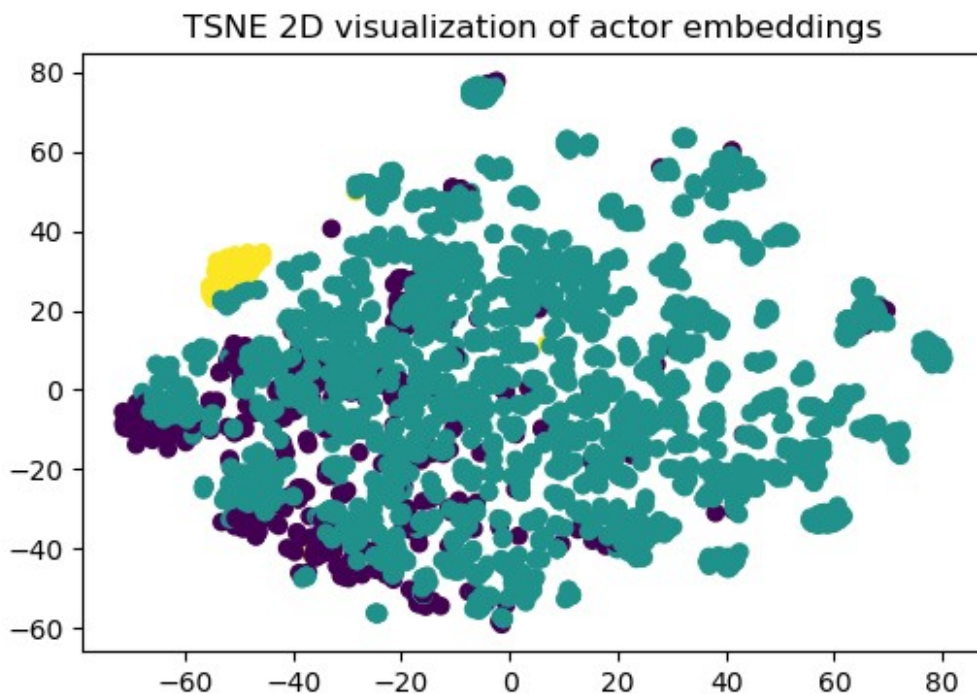
```

trans = transform(n_components=2)
actor_embeddings_2d = trans.fit_transform(actor_embeddings)

import matplotlib.pyplot as plt
plt.figure(figsize=(6, 4))
#https://stackoverflow.com/questions/28227340/kmeans-scatter-plot-plot-different-colors-per-cluster
plt.scatter(actor_embeddings_2d[:,0], actor_embeddings_2d[:,1],
c=model.labels_.astype(float))
plt.title('TSNE 2D visualization of actor embeddings')

Text(0.5,1,'TSNE 2D visualization of actor embeddings')

```



```

def cost_2(graph,cluster):
    degree=graph.degree()
    degree_value=0 # to store sum of all degree of actor nodes
    act_nodes=0 #to store sum of all unique actor nodes
    for i in degree:
        if "m" in i[0]:
            degree_value=degree_value+i[1]
        else:
            act_nodes=act_nodes+1
    value=((1/cluster)*(degree_value/act_nodes))
    return value

cost_value={}
for clu in [3,5,10,50,200,300]:
    cost1=0
    cost2=0

```

```

label=[]
algo = KMeans(n_clusters=clu)
algo.fit(movie_embeddings) # FITTING WITH KMEANS ALGO
label=algo.labels_
for i in range(clu):
    G1=nx.Graph() # drawing graph for each cluster
    label_divi=[]
    k=[index for index, value in enumerate(label) if value == i]#
    accessing each cluster based on labels_
    label_divi=[ movie_nodes[l] for l in k]
    for node in label_divi:
        sub_graph1=nx.ego_graph(B,node)
        G1.add_nodes_from(sub_graph1.nodes) # adding nodes
        G1.add_edges_from(sub_graph1.edges()) # adding edges
    cst1=cost_1(G1,clu)
    cost1=cost1+cst1 # calculating cost functions
    cst2=cost_2(G1,clu)
    cost2=cost2+cst2
    value=cost1*cost2
    cost_value[clu]=value

optimal_cluster_number = max(cost_value, key=cost_value.get)
print("optimal cluster number =",optimal_cluster_number)
#https://www.geeksforgeeks.org/python-get-key-with-maximum-value-in-dictionary/

optimal_cluster_number = 5

model= KMeans(n_clusters=optimal_cluster_number)
model.fit(movie_embeddings)
label=model.labels_

node_cluster={}
for i in range(optimal_cluster_number):
    k=[index for index, value in enumerate(label) if value == i]
    label_divi=[ movie_nodes[l] for l in k]
    for node in label_divi:
        node_cluster[node]=i

node_cluster['m858']

1

from sklearn.manifold import TSNE
transform = TSNE #PCA
trans = transform(n_components=2)
movie_embeddings_2d = trans.fit_transform(movie_embeddings)

import matplotlib.pyplot as plt
# Visualize it:
plt.figure(figsize=(6, 4))

```

```
plt.scatter(movie_embeddings_2d[:,0], movie_embeddings_2d[:,1],  
            c=model.labels_.astype(float))  
plt.title('TSNE 2D visualization of movie embeddings')  
Text(0.5,1,'TSNE 2D visualization of movie embeddings')
```

