In [1]:

```
import twitter
```

In [2]:

```
def oauth_login():
    CONSUMER_KEY = 'NPZbW3Praoi6hm4iMGZieufxF'
    CONSUMER_SECRET = 'cYTtV54sd9MQwTZk8lbsbMpxLXrH0UIaDYz0XA2o
    OAUTH_TOKEN = '839530050-cPlBOP7fHKEq4sSc03MqH2AVBaIDNTwMxf
    OAUTH_TOKEN_SECRET = '3YJtuuGkQDMZzR9zzYMLN39ATCxOMz0b15fTo
    auth = twitter.oauth.OAuth(OAUTH_TOKEN, OAUTH_TOKEN_SECRET,
    twitter_api = twitter.Twitter(auth=auth)
    return twitter_api
twitter_api = oauth_login()
print(twitter_api)
```

```
<twitter.api.Twitter object at 0x00000225BB369C18>
```

In [3]:

```
import sys
import time
from urllib.error import URLError
from http.client import BadStatusLine
import json
import twitter
```

In [4]:

```
def make_twitter_request(twitter_api_func, max_errors=10, *args

    def handle_twitter_http_error(e, wait_period=2, sleep_when_

        if wait_period > 3600: # Seconds
            print('Too many retries. Quitting.', file=sys.stder
            raise e
            if e.e.code == 401:
                print('Encountered 401 Error (Not Authorized)',
                return None
        elif e.e.code == 404:
            printnt('Encountered 404 Error (Not Found)', file=s
            return None
            elif e.e.code == 429:
                print('Encountered 429 Error (Rate Limit Exceeded)'
                if sleep_when_rate_limited:
                    print("Retrying in 15 minutes...ZzZ...", file=s
                    sys.stderr.flush()
                    time.sleep(60*15 + 5)
                    print('...ZzZ...Awake now and trying again.', f
                    return 2
                else:
                    raise e # Caller must handle the rate limiting
            elif e.e.code in (500, 502, 503, 504):
                print('Encountered {0} Error. Retrying in {1} secon
                time.sleep(wait_period)
                wait_period *= 3
                return wait_period
            else:
                raise e

        wait_period = 3
        error_count = 0

        while True:
            try:
                return twitter_api_func(*args, **kw)
            except twitter.api.TwitterHTTPError as e:
                error_count = 0
                wait_period = handle_twitter_http_error(e, wait_per
                if wait_period is None:
                    return
            except URLError as e:
                error_count += 1
                time.sleep(wait_period)
                wait_period *= 3
                print("URLError encountered. Continuing.", file=sys
                if error_count > max_errors:
                    print("Too many consecutive errors...bailing ou
                    raise
            except BadStatusLine as e:
                error_count += 1
                time.sleep(wait_period)
                wait_period *= 3
                print("BadStatusLine encountered. Continuing.", fil
                if error_count > max_errors:
                    print("Too many consecutive errors...bailing ou
                    raise
```

In [5]:

```python
from functools import partial
from sys import maxsize as maxint
```

In [6]:

```python
def get_friends_followers_ids(twitter_api, screen_name=None, us
                              friends_limit=maxint, followers_l
    #Must have either screen_name or user_id (logical xor)
    assert (screen_name != None) != (user_id != None),     "Mus
    #5000 friends and follower ids
    get_friends_ids = partial(make_twitter_request, twitter_api
                              count=5000)
    get_followers_ids = partial(make_twitter_request, twitter_a
                                count=5000)

    friends_ids, followers_ids = [], []
    #api call to get friends and follower ids using partial
    for twitter_api_func, limit, ids, label in [
                [get_friends_ids, friends_limit, friends_id
                [get_followers_ids, followers_limit, follow
            ]:

        if limit == 0: continue

        cursor = -1
        while cursor != 0:

            # Use make_twitter_request via the partially bound
            if screen_name:
                response = twitter_api_func(screen_name=screen_
            else: # user_id
                response = twitter_api_func(user_id=user_id, cu

            if response is not None:
                ids += response['ids']
                cursor = response['next_cursor']

            print('Fetched {0} total {1} ids for {2}'.format(le
                                                              la

            # XXX: You may want to store data during each itera
            # an additional layer of protection from exceptiona
```

```
                # an additional layer of protection from exceptiona

            if len(ids) >= limit or response is None:
                break

    # Do something useful with the IDs, like store them to disk
    #return no of friends and follower ids upto the limit asked
    return friends_ids[:friends_limit], followers_ids[:follower

# Sample usage

twitter_api = oauth_login()

friends_ids, followers_ids = get_friends_followers_ids(twitter_
                                                       screen_n
                                                       friends_
                                                       follower

print(friends_ids)
print(followers_ids)
```

```
Encountered 429 Error (Rate Limit Exceeded)
Retrying in 15 minutes...ZzZ...
...ZzZ...Awake now and trying again.
Fetched 330 total friends ids for sundarpichai

[78941611, 16847211, 2499289070, 22513243, 13418072, 2
464809181, 994271837361061888, 20753077, 139876086, 21
5952307]
[85816369, 1098810906127958016, 1103006578578341893, 1
103005935482613762, 1102952771131711488, 1103005596549
287938, 317698831, 749686724834500608, 27714628, 24272
67151]

Fetched 5000 total followers ids for sundarpichai
```

In [7]:

```
screen_name = 'sundarpichai'
```

In [8]:

```
response = make_twitter_request(twitter_api.friends.ids,
                                screen_name=screen_name, count
friends = response["ids"]
```

In [9]:

```python
#Finding reciprocal of friends
reciprocal_friends = set(friends)

reciprocal_friends
```

Out[9]:

```
{12,
 13,
 20,
 291,
 422,
 586,
 785,
 953,
 989,
 1081,
 1186,
 1605,
 3475,
 5017,
 5699,
 7698,
 10078,
 11113,
```

In [10]:

```python
def get_user_profile(twitter_api, screen_names=None, user_ids=N
    assert (screen_names != None) != (user_ids != None),     "M

    items_to_info = {}

    items = screen_names or user_ids

    while len(items) > 0:

        # Process 100 items at a time per the API specification
        # See http://bit.ly/2Gcjfzr for details.

        items_str = ','.join([str(item) for item in items[:100]
        items = items[100:]

        if screen_names:
            response = make_twitter_request(twitter_api.users.l
                                            screen_name=items_s
        else: # user_ids
            response = make_twitter_request(twitter_api.users.l
                                            user_id=items_str)

        for user_info in response:
            if screen_names:
                items_to_info[user_info['screen_name']] = user_
            else: # user_ids
                items_to_info[user_info['id']] = user_info

    return items_to_info

# Sample usage

twitter_api = oauth_login()

print(get_user_profile(twitter_api, screen_names=["SocialWebMin
```

```
{'SocialWebMining': {'id': 132373965, 'id_str': '13237
3965', 'name': 'MiningTheSocialWeb', 'screen_name': 'S
ocialWebMining', 'location': '', 'description': 'Get t
he source code at GitHub: http://t.co/U0VmWrXpB9',
(http://t.co/U0VmWrXpB9',) 'url': 'http://t.co/CJfJDyM
6ki', 'entities': {'url': {'urls': [{'url': 'http://t.
co/CJfJDyM6ki', 'expanded_url': 'http://miningthesocia
lweb.com', 'display_url': 'miningthesocialweb.com', 'i
ndices': [0, 22]}]}, 'description': {'urls': [{'url':
'http://t.co/U0VmWrXpB9', 'expanded_url': 'http://bit.
ly/MiningTheSocialWeb2E', 'display_url': 'bit.ly/Minin
gTheSocia…', 'indices': [31, 53]}]}}, 'protected': Fal
se, 'followers_count': 4336, 'friends_count': 0, 'list
ed_count': 221, 'created_at': 'Tue Apr 13 02:10:40 +00
00 2010', 'favourites_count': 35, 'utc_offset': None,
'time_zone': None, 'geo_enabled': False, 'verified': F
alse, 'statuses_count': 779, 'lang': 'en', 'status': {

'created_at': 'Mon Jan 28 14:06:01 +0000 2019', 'id':
1089887323116969985, 'id_str': '1089887323116969985',
'text': 'What did it take to write the new edition? We
ll, trying to keep up with a changing social media lan
dscape, for one.… https://t.co/vsaR6B4smZ',
(https://t.co/vsaR6B4smZ',) 'truncated': True, 'entiti
es': {'hashtags': [], 'symbols': [], 'user_mentions':
```

```
[], 'urls': [{'url': 'https://t.co/vsaR6B4smZ', 'expan
ded_url': 'https://twitter.com/i/web/status/1089887323
116969985', 'display_url': 'twitter.com/i/web/status/1
…', 'indices': [117, 140]}]}, 'source': '<a href="http
s://buffer.com" rel="nofollow">Buffer</a>', 'in_reply_
to_status_id': None, 'in_reply_to_status_id_str': None
, 'in_reply_to_user_id': None, 'in_reply_to_user_id_st
r': None, 'in_reply_to_screen_name': None, 'geo': None
, 'coordinates': None, 'place': None, 'contributors':
None, 'is_quote_status': False, 'retweet_count': 2, 'f
avorite_count': 9, 'favorited': False, 'retweeted': Fa
lse, 'possibly_sensitive': False, 'lang': 'en'}, 'cont
ributors_enabled': False, 'is_translator': False, 'is_
translation_enabled': False, 'profile_background_color
': '352726', 'profile_background_image_url': 'http://a
bs.twimg.com/images/themes/theme5/bg.gif', 'profile_ba
ckground_image_url_https': 'https://abs.twimg.com/imag
es/themes/theme5/bg.gif', 'profile_background_tile': F
alse, 'profile_image_url': 'http://pbs.twimg.com/profi
le_images/1154493071/Picture_7_normal.png', 'profile_i
mage_url_https': 'https://pbs.twimg.com/profile_images
/1154493071/Picture_7_normal.png', 'profile_link_color
': 'D02B55', 'profile_sidebar_border_color': '829D5E',
'profile_sidebar_fill_color': '99CC33', 'profile_text_
color': '3E4415', 'profile_use_background_image': True
, 'has_extended_profile': False, 'default_profile': Fa
lse, 'default_profile_image': False, 'following': Fals
e, 'follow_request_sent': False, 'notifications': Fals
e, 'translator_type': 'none'}, 'sundarpichai': {'id':
14130366, 'id_str': '14130366', 'name': 'Sundar Pichai
', 'screen_name': 'sundarpichai', 'location': '', 'des
cription': 'CEO, Google', 'url': None, 'entities': {'d
escription': {'urls': []}}, 'protected': False, 'follo
wers_count': 2186540, 'friends_count': 330, 'listed_co
unt': 6066, 'created_at': 'Wed Mar 12 05:51:53 +0000 2
008', 'favourites_count': 738, 'utc_offset': None, 'ti
me_zone': None, 'geo_enabled': True, 'verified': True,
'statuses_count': 1194, 'lang': 'en', 'status': {'crea
ted_at': 'Sat Mar 02 04:44:40 +0000 2019', 'id': 11017
04852592254976, 'id_str': '1101704852592254976', 'text
': 'RT @SusanWojcicki: Great to see so many familiar @
YouTube @Google faces today at the #LWTSummit! https:/
/t.co/JUdARgkVsH', (https://t.co/JUdARgkVsH',)
'truncated': False, 'entities': {'hashtags': [{'text':
'LWTSummit', 'indices': [85, 95]}], 'symbols': [], 'us
```

er_mentions': [{'screen_name': 'SusanWojcicki', 'name'
: 'Susan Wojcicki', 'id': 15828408, 'id_str': '1582840
8', 'indices': [3, 17]}, {'screen_name': 'YouTube', 'n
ame': 'YouTube', 'id': 10228272, 'id_str': '10228272',
'indices': [49, 57]}, {'screen_name': 'Google', 'name'
: 'Google', 'id': 20536157, 'id_str': '20536157', 'ind
ices': [58, 65]}], 'urls': [], 'media': [{'id': 110167
2132273102850, 'id_str': '1101672132273102850', 'indic
es': [97, 120], 'media_url': 'http://pbs.twimg.com/med
ia/D0ns96cV4AIhcIX.jpg', 'media_url_https': 'https://p
bs.twimg.com/media/D0ns96cV4AIhcIX.jpg', 'url': 'https
://t.co/JUdARgkVsH', 'display_url': 'pic.twitter.com/J
UdARgkVsH', 'expanded_url': 'https://twitter.com/Susan
Wojcicki/status/1101672146621759488/photo/1', 'type':
'photo', 'sizes': {'thumb': {'w': 150, 'h': 150, 'resi
ze': 'crop'}, 'large': {'w': 2048, 'h': 1536, 'resize'
: 'fit'}, 'medium': {'w': 1200, 'h': 900, 'resize': 'f
it'}, 'small': {'w': 680, 'h': 510, 'resize': 'fit'}},
'source_status_id': 1101672146621759488, 'source_statu
s_id_str': '1101672146621759488', 'source_user_id': 15
828408, 'source_user_id_str': '15828408'}]}, 'extended
_entities': {'media': [{'id': 1101672132273102850, 'id
_str': '1101672132273102850', 'indices': [97, 120], 'm
edia_url': 'http://pbs.twimg.com/media/D0ns96cV4AIhcIX
.jpg', 'media_url_https': 'https://pbs.twimg.com/media
/D0ns96cV4AIhcIX.jpg', 'url': 'https://t.co/JUdARgkVsH
', 'display_url': 'pic.twitter.com/JUdARgkVsH', 'expan
ded_url': 'https://twitter.com/SusanWojcicki/status/11
01672146621759488/photo/1', 'type': 'photo', 'sizes':
{'thumb': {'w': 150, 'h': 150, 'resize': 'crop'}, 'lar
ge': {'w': 2048, 'h': 1536, 'resize': 'fit'}, 'medium'
: {'w': 1200, 'h': 900, 'resize': 'fit'}, 'small': {'w
': 680, 'h': 510, 'resize': 'fit'}}, 'source_status_id
': 1101672146621759488, 'source_status_id_str': '11016
72146621759488', 'source_user_id': 15828408, 'source_u
ser_id_str': '15828408'}]}, 'source': '<a href="http:/
/twitter.com/download/android" rel="nofollow">Twitter

for Android</a>', 'in_reply_to_status_id': None, 'in_r
eply_to_status_id_str': None, 'in_reply_to_user_id': N
one, 'in_reply_to_user_id_str': None, 'in_reply_to_scr
een_name': None, 'geo': None, 'coordinates': None, 'pl
ace': None, 'contributors': None, 'retweeted_status':
{'created_at': 'Sat Mar 02 02:34:42 +0000 2019', 'id':
1101672146621759488, 'id_str': '1101672146621759488',

```
'text': 'Great to see so many familiar @YouTube @Googl
e faces today at the #LWTSummit! https://t.co/JUdARgkV
sH', (https://t.co/JUdARgkVsH',) 'truncated': False, '
entities': {'hashtags': [{'text': 'LWTSummit', 'indice
s': [66, 76]}], 'symbols': [], 'user_mentions': [{'scr
een_name': 'YouTube', 'name': 'YouTube', 'id': 1022827
2, 'id_str': '10228272', 'indices': [30, 38]}, {'scree
n_name': 'Google', 'name': 'Google', 'id': 20536157, '
id_str': '20536157', 'indices': [39, 46]}], 'urls': []
, 'media': [{'id': 1101672132273102850, 'id_str': '110
1672132273102850', 'indices': [78, 101], 'media_url':
'http://pbs.twimg.com/media/D0ns96cV4AIhcIX.jpg', 'med
ia_url_https': 'https://pbs.twimg.com/media/D0ns96cV4A
IhcIX.jpg', 'url': 'https://t.co/JUdARgkVsH', 'display
_url': 'pic.twitter.com/JUdARgkVsH', 'expanded_url': '
https://twitter.com/SusanWojcicki/status/1101672146621
759488/photo/1', 'type': 'photo', 'sizes': {'thumb': {
'w': 150, 'h': 150, 'resize': 'crop'}, 'large': {'w':
2048, 'h': 1536, 'resize': 'fit'}, 'medium': {'w': 120
0, 'h': 900, 'resize': 'fit'}, 'small': {'w': 680, 'h'
: 510, 'resize': 'fit'}}}]}, 'extended_entities': {'me
dia': [{'id': 1101672132273102850, 'id_str': '11016721
32273102850', 'indices': [78, 101], 'media_url': 'http
://pbs.twimg.com/media/D0ns96cV4AIhcIX.jpg', 'media_ur
l_https': 'https://pbs.twimg.com/media/D0ns96cV4AIhcIX
.jpg', 'url': 'https://t.co/JUdARgkVsH', 'display_url'
: 'pic.twitter.com/JUdARgkVsH', 'expanded_url': 'https
://twitter.com/SusanWojcicki/status/110167214662175948
8/photo/1', 'type': 'photo', 'sizes': {'thumb': {'w':
150, 'h': 150, 'resize': 'crop'}, 'large': {'w': 2048,
'h': 1536, 'resize': 'fit'}, 'medium': {'w': 1200, 'h'
: 900, 'resize': 'fit'}, 'small': {'w': 680, 'h': 510,
'resize': 'fit'}}}]}, 'source': '<a href="http://twitt
er.com/download/iphone" rel="nofollow">Twitter for iPh
one</a>', 'in_reply_to_status_id': None, 'in_reply_to_
status_id_str': None, 'in_reply_to_user_id': None, 'in
_reply_to_user_id_str': None, 'in_reply_to_screen_name

': None, 'geo': None, 'coordinates': None, 'place': No
ne, 'contributors': None, 'is_quote_status': False, 'r
etweet_count': 48, 'favorite_count': 789, 'favorited':
False, 'retweeted': False, 'possibly_sensitive': False
, 'lang': 'en'}, 'is_quote_status': False, 'retweet_co
unt': 48, 'favorite_count': 0, 'favorited': False, 're
tweeted': False, 'possibly_sensitive': False, 'lang':
```

'en'}, 'contributors_enabled': False, 'is_translator':
False, 'is_translation_enabled': False, 'profile_backg
round_color': '1A1B1F', 'profile_background_image_url'
: 'http://abs.twimg.com/images/themes/theme9/bg.gif',
'profile_background_image_url_https': 'https://abs.twi
mg.com/images/themes/theme9/bg.gif', 'profile_backgrou
nd_tile': False, 'profile_image_url': 'http://pbs.twim
g.com/profile_images/864282616597405701/M-FEJMZ0_norma
l.jpg', 'profile_image_url_https': 'https://pbs.twimg.
com/profile_images/864282616597405701/M-FEJMZ0_normal.
jpg', 'profile_link_color': '2FC2EF', 'profile_sidebar
_border_color': '181A1E', 'profile_sidebar_fill_color'
: '252429', 'profile_text_color': '666666', 'profile_u
se_background_image': True, 'has_extended_profile': Fa
lse, 'default_profile': False, 'default_profile_image'
: False, 'following': True, 'follow_request_sent': Fal
se, 'notifications': False, 'translator_type': 'none'}

In [11]:

```python
import pandas as pd
df = pd.DataFrame(columns=['ID','ReciprocalFriend'])
df.to_csv('ReciprocalFriend.csv', index=False)

# Our function
def save_followers(fid, reciprocal_friend):
    data_frame_rf = [[str(fid), str(i)] for i in reciprocal_fri
    #print(data_frame_rf)
    df = pd.DataFrame(data_frame_rf, columns=['ID','ReciprocalF
    with open('ReciprocalFriend.csv', 'a') as f:
        df.to_csv(f,header=False, index=False)
```

In [12]:

```
def crawl_followers(twitter_api, screen_name, limit=1000000, de

    # Resolve the ID for screen_name and start working with IDs
    seed_id = str(twitter_api.users.show(screen_name=screen_nam
    friends_ids, followers_ids = get_friends_followers_ids(twit
                                 friends_limit=limit, followers
    rp_friend = list(set(friends_ids) & set(followers_ids))
    top_five = get_user_profile(twitter_api, user_ids=rp_friend
    next_queue = top_five
    # Store a seed_id => _follower_ids mapping in MongoDB

    save_followers(seed_id, next_queue)

    d = 1
    # Note that in the example in the next cell,
    # we never enter this loop.
    while d < depth:
        print("Number of ", d,"- Distance node", len(next_queue
        d += 1
        # Reset the next_queue so that we can
        # start building up the next level
        # of followers-of-followers
        (queue, next_queue) = (next_queue, [])
        # Loop through the current
        # level of followers
        for fid in queue:
            friends_ids, followers_ids = get_friends_followers_
                                    friends_limit=limit, followers_
            # Store an ID with a string recording
            # IDs of followers of the user with ID "fid"
            rp_friend = list(set(friends_ids) & set(followers_i
            if (len(rp_friend) == 0):
                continue
            top_five = get_user_profile(twitter_api, user_ids=r
            save_followers(str(fid), top_five)
            # Extending the list
            next_queue += top_five
```

In [13]:

```
screen_name = 'TechnologyGuy'
crawl_followers(twitter_api, screen_name, depth=2, limit=5000)
```

```
Fetched 5000 total friends ids for 27142322
Fetched 5000 total followers ids for 27142322

Number of  1 - Distance node 100

Fetched 36 total friends ids for 1058014853657436160
Fetched 11 total followers ids for 1058014853657436160
Fetched 965 total friends ids for 982212114398904321
Fetched 254 total followers ids for 982212114398904321
Fetched 3877 total friends ids for 957273470269841410
Fetched 1483 total followers ids for 957273470269841410
Fetched 4996 total friends ids for 839297193664184321
Fetched 1804 total followers ids for 839297193664184321
Fetched 5000 total friends ids for 801677550191640580
Fetched 5000 total followers ids for 801677550191640580
Fetched 1004 total friends ids for 1068482784673652741
Fetched 125 total followers ids for 1068482784673652741
```

In [ ]:

```

```

In [14]:

```
import numpy as np
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
# Create Graph from file
df = pd.read_csv("ReciprocalFriend.csv")
x_point = list(df[df.columns[0]].values)
y_point = list(df[df.columns[1]].values)
edges_list = []
for i in range(len(x_point)):
    edges_list.append((x_point[i], y_point[i]))
node_list = set(x_point+y_point)
RG = nx.Graph()
RG.add_nodes_from(node_list)
RG.add_edges_from(edges_list)
# Display some graph information such as number of nodes and ed
print ("Number of Nodes :", RG.number_of_nodes())
print ("Numebr of edges :", RG.number_of_edges())

```

```
Number of Nodes : 3165
Numebr of edges : 3342
```

In [20]:

```
lmbds,vctrs = np.linalg.eig(L)
indx = [i for i in range(len(lmbds)) if lmbds[i] > .01 and lmbd
RG_mbd = vctrs[:,indx]
print ("Number of Communities:", len(indx))
```

```
Number of Communities: 5
```

In [21]:

```
est = KMeans(max_iter = 100000, n_clusters = len(indx), n_init
results_df['kmeans'] = est.fit(RG_mbd)
# y_pred[i] = ada.predict([X.iloc[i, :]])[0]
# Apply k-means
# est = KMeans(n_clusters=len(indx))
# est.fit(RG_mbd)
```

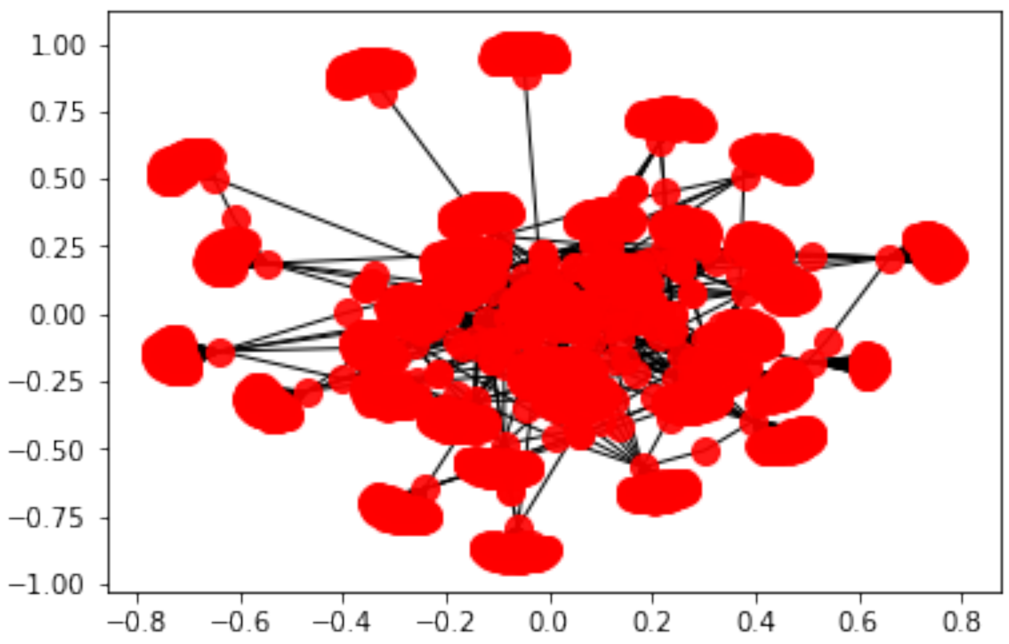
```
---------------------------------------------------------------------------
ComplexWarning                            Traceback (m
ost recent call last)
D:\Anaconda\lib\site-packages\sklearn\utils\validation
.py in check_array(array, accept_sparse, accept_large_
sparse, dtype, order, copy, force_all_finite, ensure_2
d, allow_nd, ensure_min_samples, ensure_min_features,
warn_on_dtype, estimator)
    526                 warnings.simplefilter('error',
ComplexWarning)
--> 527                 array = np.asarray(array,
dtype=dtype, order=order)
    528             except ComplexWarning:

D:\Anaconda\lib\site-packages\numpy\core\numeric.py in
asarray(a, dtype, order)
    500     """
--> 501     return array(a, dtype, copy=False, order=o
```

In [23]:

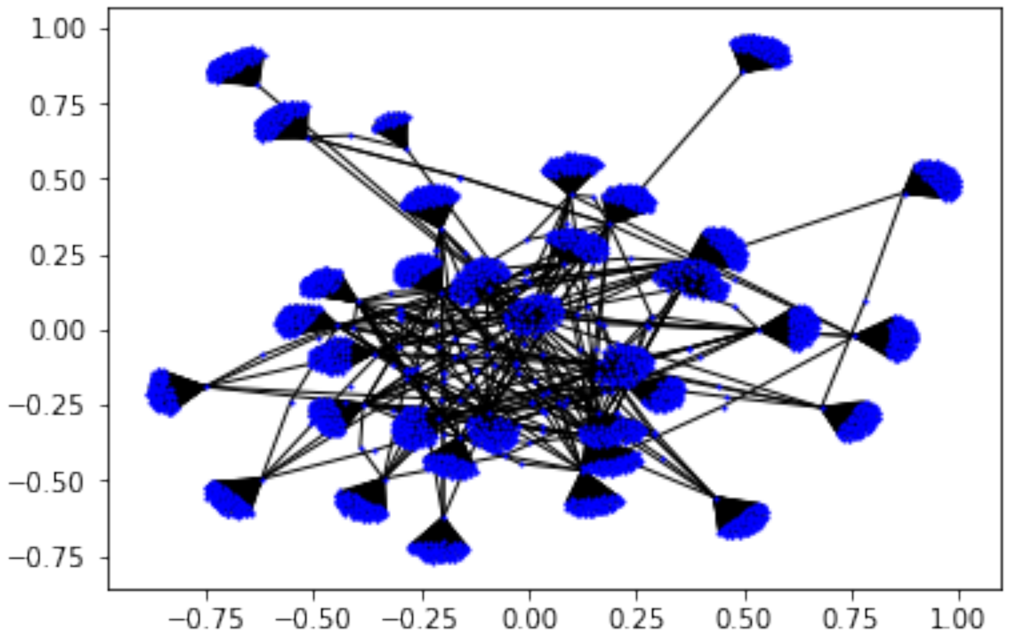
```
pos = nx.spring_layout(RG)
nx.draw_networkx_nodes(RG,pos,
                              nodelist=node_list,
                              node_color='r',
                              node_size=100,
                         alpha=0.9)
nx.draw_networkx_edges(RG,pos)
plt.show()
```

```
D:\Anaconda\lib\site-packages\networkx\drawing\nx_pyla
b.py:611: MatplotlibDeprecationWarning: isinstance(...
, numbers.Number)
  if cb.is_numlike(alpha):
```



In [24]:

```
pos = nx.spring_layout(RG)
nx.draw_networkx_nodes(RG,pos,nodelist=node_list, node_color='b
                               node_shape='o', node_size=1, alpha=1)
nx.draw_networkx_edges(RG,pos)
plt.show()
```



In [25]:

```
print('The diameter is: ',nx.diameter(RG))
```

```
The diameter is:  4
```

In [26]:

```
print('The average distance between nodes is: ',nx.average_shor
```

```
The average distance between nodes is:  3.870863166388
0584
```