Wesley Swedenburg
CS 491

Homework 2

repo : https://github.com/mrpoodestump/cs491HW2
The python file that was used was createDataBase.py
The database is KevinsNetwork.db

1. Collecting the data.

One of my friends is a local comedian who has a small following on twitter that comprises many other local comedians. I thought this friend would be a good seed user to find a network of local comedians in Albuquerque.

Initially, for each user I would gather all their followers and randomly choose one then gather all their followers which gave me a ton of extra users with only one 'following' edge to each random user I had gathered. I then filtered out the one edge followers which left me with a nicely connected graph of local comedians and followers of local comedians. Because I was randomly walking from an initial user and had to stop and start walking from that initial user multiple times due to rate limits and programming errors, the network is biased towards the initial user, for this reason I exclude that user from the top 5 page ranked users. Below is 656 nodes/users.

Figure 1: from Giphy
The graph:

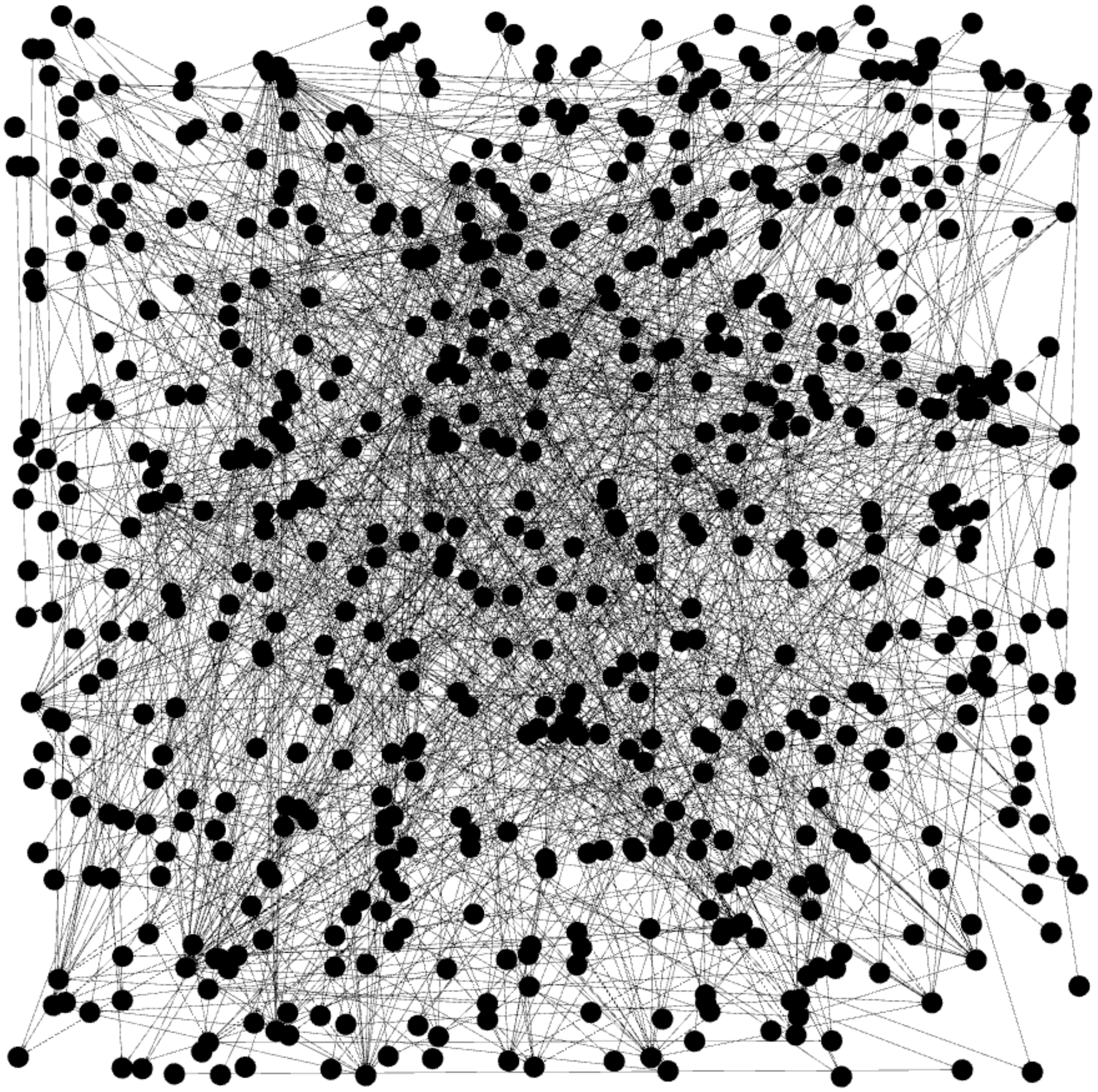
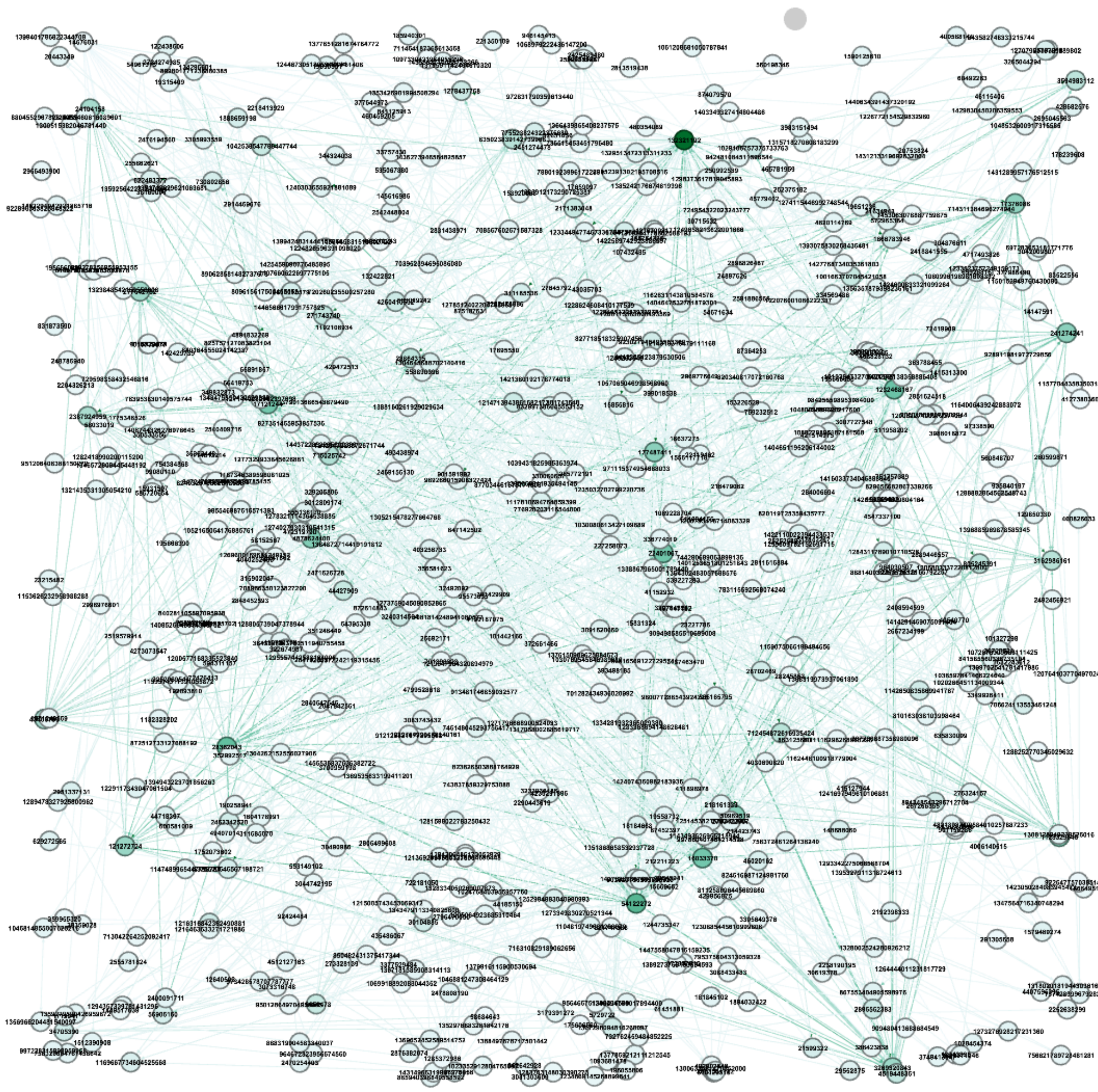
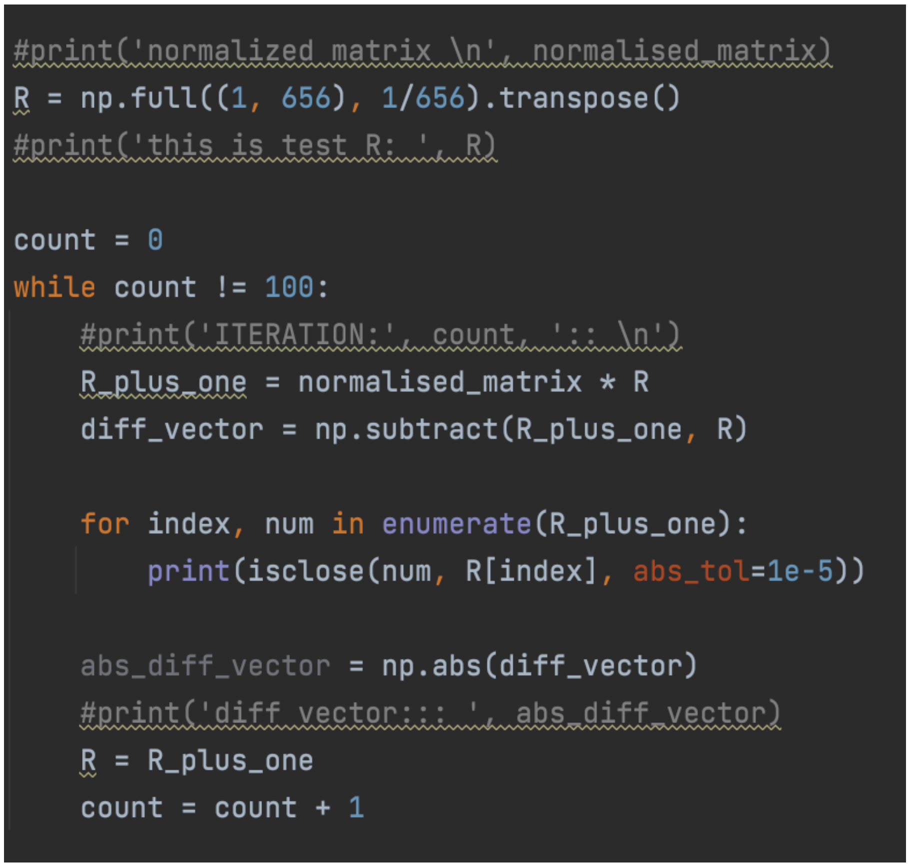


Figure 2:
The graph with user ids as labels and the darker green the node is the more followers.



2. Page Rank

After running page rank the top results were expected, 3 of the top results were local comedians with many followers the other 2 were local theatre organizations. Using python’s built-in isclose function that returns true or false if a number in the previous iteration and the current iteration are close enough to a specified decimal place. After trial and error it seems like things were converging nicely around 100 iterations, although it depended on how precise I wanted the values. Below is the while loop in the script that iterates the page rank vector.



```python
#print('normalized matrix \n', normalised_matrix)
R = np.full((1, 656), 1/656).transpose()
#print('this is test R: ', R)

count = 0
while count != 100:
    #print('ITERATION:', count, ':: \n')
    R_plus_one = normalised_matrix * R
    diff_vector = np.subtract(R_plus_one, R)

    for index, num in enumerate(R_plus_one):
        print(isclose(num, R[index], abs_tol=1e-5))

    abs_diff_vector = np.abs(diff_vector)
    #print('diff vector::: ', abs_diff_vector)
    R = R_plus_one
    count = count + 1
```