EE 232E
Graphs and Network Flows
Final Project
Winter 2016

Liqiang Yu, Rongjing Bai, Yunwen Zhu
904592975, 204587519, 104593417

05-28-2016

Contents

1 Problem1

We use python to parse the data in actor_movies.txt and actress_movies.txt to merge the information about actor/actress. And when parsing the data, we remove those with less than five movies. Also, we create actor_id parameter on basis of the order of actor/actress for convenience in mapping for the name. To help mapping with actor_id and actor's name, we write actor_id.py and create the correspoinding mapping actor_id.txt. Finally, there are totally 244300 actor/actress in the merge_actor.txt and the corresponding parsing program is merge_actor.py.

2 Problem2

In this problem, we create the weighted directed graph based on the formula $\frac{|Si \cap Sj|}{|Si|}$, where Si = $\{m \mid i \in V$,m is a movie in which i has acted$\}$. We first create a dictionary movie_actor with the movie as the key and the actors that participate in this movie as the value for quick search. Then we create another dictionary edge_weight with the two actor_ids as the key and the number of their same movies as the value. Thus, by traversing each movie with more than two actors involved, we create the edge according to the edge_weight with the two actor as the vertices and calculated weight. Finally, there are 58026910 edges in this graph in edge_weight.txt and the corresponding program is create_graph.py. Also, note that there are many directed edges with weight = 1, that means the movies of that actor totally intersected in the movies of another actor, which fit well in reality as well as the data.

3 Problem 3

We run the pagerank algorithm with damping = 0.85 on our weighted directed graph and we get the names of those in top 10 pageranks with our actor_id.txt. The names are listed in Table 1.

From the results of Table 1, we just know the name of a few of them, but these top 10 are certainly not the famous stars. Thus, we list the names of famous stars in our opinion and then calculate their corresponding pagerank. Their names and pageranks are listed in Table 2.

From the Table 2, out of expection, the famous stars we choose are all have a relatively low pagerank, especially when compared with those top 10 pageranks.There are many reasons for this phenomenon that we are not familar with those high pagerank scores. First, due to our subjectivity and our time and country, we may not actually know certain movie stars from other countries but still be well-known in their country. Or these movie stars were really famous in their times but are not sound familar to our generation. Secondly, high

Table 1: names of actors with top 10 pageranks

Flowers, Bess	0.0001580023
Tatasciore, Fred	0.0001377731
Blum, Steve (IX)	0.0001340366
Harris, Sam (II)	0.0001333384
Jeremy, Ron	0.0001270027
Miller, Harold (I)	0.0001165205
Lowenthal, Yuri	0.0001090534
Downes, Robin Atkin	0.000104297
Phelps, Lee (I)	0.0001012679
Sayre, Jeffrey	0.0001006508

Table 2: names and pageranks of 10 famous stars

Page, Ellen	1.172932e-05
Portman, Natalie	2.365215e-05
Linney, Laura	1.848715e-05
Rudd, Paul (I)	2.888519e-05
Dennings, Kat	9.740525e-06
Lively, Blake	3.248758e-05
Clooney, George	0.0001090534
Reynolds, Ryan (I)	2.143052e-05
Hanks, Colin	1.475533e-05
Radcliffe, Daniel	1.430697e-05

rankpage score may not mean great reputation. Those not famous actors tend to try in every kind of movie and act in many movies just as a supporting role. Thus, they do have a closer relationship and connection to many other actors then have a high pagerank score.

In order to find the 10 most significant pairs, we convert our original directed graph into the undirected one with weight as the mean of the directed edges. Then we sort the weight of all edges, and find out the top 10 as our significant pairs. After that, we use our actor_id.txt to map into their corresponding names. The names of signicicant pairs are shown in the Table 3. From the Table 3, we

Table 3: 10 significant pairs

Mounir, Dean	Bailey, Carla Jo
Kim, Su-hyeon (II)	Phillipson, Caspar
Harvey, Nita	Gamal, Mimi
Perlmutter, Kalle	Evans, Kahshanna
Stefancyk, Bryan	de Carvalho, Inalda
Matthews, Liam (III)	DuChesne, Derek
Beltrn, Pedro (I)	McNicol, Valerie (III)
Kasten, Kristina (II)	Gomide, Tnia
Wagner, David (I)	Zhao, Jun (I)
Boulanger, Vronique	Carson, Lisa Nicole

can see that these pairs may occur between different actors and actresses, but these actors all have relatively high weight with their connected edges. That means that they have collaborated with many of other actors in most of their participating movies especially to those whose movies are totally intersected with other actors. However, we actually barely know their names and they do not appear in the top pagerank as well. Actually, their page ranks are relatively low compared with our top pageranks. As, the pageranks do not simplycount for the simple two vertices'links and they consider for the total connectivity.

As for the reason why the names in the pairs collaborate in so many movies but we do not know well, there are maybe two of them from our assumption. First, maybe we really are not familiar with them. Actually, when we try to search their names in the Google, we find out that some of them are not our generation. And movies in that age is definitely limited in the numbers, thus actors of that age are more likely to collaborate. Also, some of the actors are from some country that we are not familar with, thus these pairs may sound pretty famous to people from their country. Secondly, these actors may always act in the same kind of movies, thus their chance of co-acting with the same group of people are relatively high.

4 Problem 4

We remove all movies with less than 5 actors/actresses in movie-list.txt file, which contains the parsed data with movie actor/actress information.
Then we map the actor to each movie and generate movie-edge-list file The jaccard index is defined as

$$J_{ij} = \frac{l_i \cap l_j}{l_i \cup l_j}$$

where l_i denotes the actor/actress list of movie i. Then we build the movie network based on the movie-edge-list and add genre, name, rating and director as the attributes of each movie node. There are 244655 nodes in the network.

5 Problem 5

After we constructed the network, there are 67 communities in total. The genres of each community occuring higher than 20% are shown in table 4, not available means there are no genres occuring higher than 20% in that community.

Table 4: The genre of each community

1	drama
2	drama
3	short
4	drama
5	drama
6	drama
7	adult
8	drama
9	short
10	drama
11	drama
12	short
13	short
14	drama
15	drama
16	drama
17	drama
18	adult
19	drama
20	comedy drama
21	drama romance
22	drama romance
23	thriller
24	short
25	short
26	romance
27	Not available
28	drama
29	short
30	drama romance
31	sci-fi thriller
32	short
33	short
34	drama
35	drama

36	short
37	horror thriller
38	thriller
39	horror short thriller
40	comedy documentary drama
41	Not available
42	comedy short
43	action short
44	comedy
45	short thriller
46	comedy romance
47	documentary
48	short
49	short
50	short
51	short
52	short
53	short
54	horror short thriller
55	short
56	drama thriller
57	sci-fi short
58	adventure thriller
59	horror short thriller
60	short
61	sci-fi short
62	adventure crime short
63	comedy short thriller
64	short
63	comedy short thriller
64	short
65	short
66	short
67	short

From the statistics, we can see that the most common tags are short and drama. I think the tags are meaningful because usually one actor/actress tends to act in one or several movies genres. So the tag should represents the genres that the actors/actresses in the lists of the community tends to act in. So we can use the genres to tag the community. However, most of the tag has frequency lower than 0.3 so the tag cannot represents the whole community.

6 Problem 6

After searching the community, we found the nearest neighbors.

Batman v Superman: Dawn of Justice (2016):
Man of Steel (2013),
Into the Storm (2014),
Real Steel (2011),
Sparkle (2012/I),
Salvation Boulevard (2011)

Mission: Impossible - Rogue Nation:
Fan (2015),
Phantom (2015)
Suffragette (2015),
Breaking the Bank (2014)
The Theory of Everything (2014)

Minions (2015)
The Lorax (2012),
Inside Out (2015),
Despicable Me 2 (2013),
Up (2009),
Surfs Up (2007)

Then we check the genres. The three movies belongs to community 3, with short as the genre. However, they are not short genres, their genres are thriller or sci-fi. I think this is because there are so many short movies that many of the communities have the tag of short.

In fact, as mentioned above, the frequency are less than 0.3 for most of the tag, so it is not weird movie genres are diffrent with the tag of the community.

7 Problem 7

To predict the ratings of the above three movies, we use both ratings from their neighbors and from nodes in the same community. The point is how to combine

them together. We tried four different methods listed below

First of all, let's assign the rates of the neighbor nodes to be R_n and the rates of the nodes in the same community to be R_c. The first method is to take the averages of the two rate vectors and then take the average of their averages.

$$rate = \frac{E(R_n) + E(R_c)}{2}$$

The second method is to take the sum of two rating vectors and then divide them by their sum of the length.

$$rate = \frac{\sum R_n + \sum R_c}{|R_n| + |R_c|}$$

The third method considers the edge weight between two nodes when calculate the mean rate of the neighbor nodes, because the edge weight can show how closely each two nodes are connected. Let's assign the edge weight vector to be E_w

$$E'(R_n) = \sum R_n \cdot E_w$$
$$rate = \frac{E'(R_n) + E(R_c)}{2}$$

The fourth method is similar to the second method, however we replace the sum to be weighted sum

$$rate = \frac{|R_n| \cdot \sum R_n \cdot E_w + \sum R_c}{|R_n| + |R_c|}$$

Table 5 shows the results of the predicting rates with four methods. From the table we can see that the ratings of the three movies tend to be between 5.5 and 6.5.

Table 5: Predicting rates with neighborhood and same-community nodes

Movie Name	Method 1	Method 2	Method 3	Method 4
Batman v Superman: Dawn of Justice (2016)	6.275912	6.191796	5.383655	6.158682
Mission: Impossible - Rogue Nation (2015)	6.211204	6.189184	5.26059	6.162731
Minions (2015)	6.512196	6.19809	6.360177	6.19361

8 Problem 8

In this part, we trained different regression models to predict the movie ratings, the features we use are top 5 actor/actress page ranks computed in Problem 3 and whether the director is in the top 100 directors list. First we fetch the directors of the "IMDB top 250 movies" and then get the unique 100 directors

as well as their rankings. So the feature vector is of length 106. The first 5 items are 5 highest page ranks, the sixth item is a boolean value of whether the director is in the top 100 directors list. If so, the remaining items show his/her ranking. The corresponding item will be 1 according to the ranking.

We trained the linear regression model and random forest regression model to predict the ratings and use 10-fold cross validation method to evaluate the results and get rid of the overfitting problem. First we randomly shuffled the whole dataset and splited it into 10 folds. Then each time we use 9 of 10 to do the training and use the remaining 1 part to do the test experiment. We check the root mean square error(RMSE) to evaluate the predicting results.

8.1 Linear Regression Model

The root mean square error of the 10-fold cross-validation experiment is shown in figure 1. The average error is 1.27417955416. The predicting rates is shown in table 6

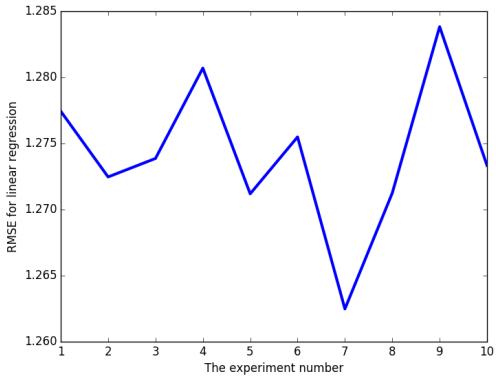

Figure 1: RMSE for the linear regression model

Table 6: Predicting rates with linear regression model

Movie Name	Predicting Rate
Batman v Superman: Dawn of Justice (2016)	6.14947715
Mission: Impossible - Rogue Nation (2015)	6.16647388
Minions (2015)	6.16990139

8.2 Random Forest Regression Model

The root mean square error of the 10-fold cross-validation experiment is shown in figure 2. The average error is 1.26341071465. The predicting rates is shown

in table 7.

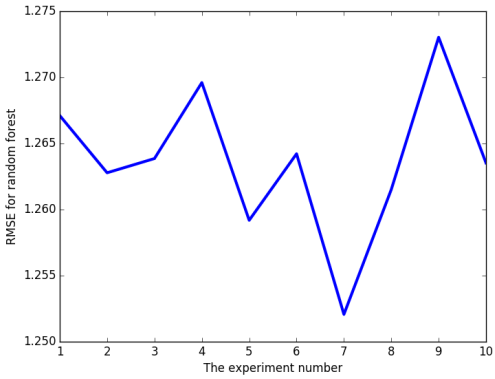

Figure 2: RMSE for the random forest regression model

Table 7: Predicting rates with random forest regression model

Movie Name	Predicting Rate
Batman v Superman: Dawn of Justice (2016)	6.09615458
Mission: Impossible - Rogue Nation (2015)	6.05967233
Minions (2015)	6.13684098

Basically the two models produced pretty similar results, which are also similar to the results in Problem 7, which means these methods are stable and accurate.

9 Problem 9

In this part we constructed a bipartite graph to predict the movie rates. There are 244300 actors and 246335 movies. Each actor/actress is connected to the movies he acted. There are 4023858 edges in the graph. We use four different methods to assign scores to actors based on the movie ratings he acted : mean, max, median and tf-idf. The first three methods are pretty self-explained, we just calculated the mean, max or median of the movie ratings as the actor score.

For tf-idf method, we refered to the tf-idf method in the information retrieval field. When assigning a score to an actor, we give each movie he acted a tf-idf weight as

$$tfidf = \frac{1}{M_i} \times log\frac{A}{A_j}$$

where M_i is the number of movies actor i acted, A is the total number of actors and A_j is the number of actors movie j has. Let's assume the normalized tf-idf

weight vector for each actor to be T, then the score for an actor is

$$score = T^2 \cdot R$$

where R is the corresponding movie ratings. Finally we take the average scores of the actors to be the ratings of the movies we want to predict.

The results of the four methods is shown in table 8. From the table we can see that mean, median and tf-idf produced similar results, so we believe these three methods are proper for predicting movie rates in the bipartite graph.

Table 8: Predicting rates with bipartite graph

Movie Name	Mean	Max	Median	TF-IDF
Batman v Superman: Dawn of Justice (2016)	6.439798	7.864211	6.495263	6.435147
Mission: Impossible - Rogue Nation (2015)	6.541897	8.018889	6.698333	6.537411
Minions (2015)	6.896637	8.686364	7.025	6.907668