EE232E Graphs and Network Flows

Spring 2017_Project2

Team Member:
Xiongfeng Hu (304753117)
Younan Liang (504759929)
Dui Lin (504759948)

1. Introduction

This project is aiming to create and explore a network from IMDb movie datasets. IMDb (http://www.imdb.com) is a popular site cataloging almost every movie ever made all around the world. The study of IMDb network could be very interesting for the following reasons: First, most people know about and can relate to movies and actors. Thus, when presented with a visualization of movie data, they will try to find their favorite movies and actors, identify movies of potential interest or explore the complex co-actor relationships among actors. Second, the dataset has rich information on each movie and actor allowing for a wide variety of data analyses. Third, the dataset is sufficiently clean (as compared to the extract useful information from cumbersome URLs) and structured so that analysis can be done without using complex semantic matching techniques.

The datasets for this project comprise **actor_movies.txt** under the format of {actor1}\t\t{movie1}\t\t{movie2}\t\t{movie3}, **actress_movies.txt** under the formation of {actress1}\t\t{movie1}\t\t{movie2}\t\t{movie3}, **director_movies.txt** under the format of {director1}\t\t{movie1}\t\t{movie2}\t\t{movie3}, **movie_genres.txt** under the format of {movie1}\t\t{genre}, **movie_rating.txt** under the format of {movie1}\t\t{rating}. Since dataset has a very large size, accessing them and retrieving relevant information by processing line by line (naive reading mechanism) would approximately take 1-day of computational time. In order to facilitate faster reading and parsing we used alternate mechanism supported by libraries like Kmisc, doMC.

2. Problem Solutions

Problem 1. Data cleaning and merging

Here we use Python to parse the data in **actor_movies.txt** & **actress_movies.txt** and merge the information about actor/actress. The process is that we first remove all actors/actresses with less than 10 movies in two txt file by removing the lines whose lengths are less than 11 (considering the name as one word), respectively. We also do data preprocessing on the actors/actresses list with less than 5 movies for future use, whose length are less than 6. Then we append actress data to actor data. Also, we create actor/actress id parameter on basis of the order of actor/actress for convenience in mapping for the name. We generate **id_actor.txt** and **actor_id.txt** to facilitate next few problems implementation like creating hashmap or dictionary. During the programming process, we encounter the problem of encoding, the Python program will continue to break down due to a unicode_parse_error at 1920 lines in actor file, so we have to throw a exception for this case. Finally, for threshold = 10, we have 74599 actors left and 38534 actresses left, so total number will be 113133; For threshold = 5, we have 155222 actors and 89080 actresses left, so total will be 244302. The data has been store as **merge_movies.txt** with the format of {act_id}\t\t{movie1}\t\t{movie2}\t\t...{movie10}....

Problem 2. Construct a weighted directed graph

After obtaining the pre-processed data, we created three dictionaries in Python: 1) Key: ID, Value: Movie; 2) Key: Movie, Value: ID 3) Key: ID, Value: [temp weight, edge_weight]. In order to speed up computation and diminish the workload for unnecessary edges. We are using three nested loop: First, we loop over actors (actor index = i); Then we loop over the movies that actor i has played in (movie index = j); Finally, we create a dictionary for every appeared actor in the upper loop, and then loop over actors in movie j (actor index = k), which means i and j participate in the same movies.

We create the weighted directed graph based on the formula $E = \{(i,j)|i,j \in V, S_i \cap S_j \neq \emptyset\}$, where weight is assigned as $\frac{|S_i \cap S_j|}{|S_i|}$ and vertices will be all actors/actresses in the previous list. Finally, there are 34267114 weighted edges appeared in this graph stored in **edge_weight.txt** and the corresponding program is **generate_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 fits well in reality as well as the data.

Problem 3. Run pagerank algorithm on the network

After constructing the IMDb actor/actress network, we calculated the pagerank of all the node with page.rank() function in igraph package. Since the vertices are numbered in the same order as the list of actors/actress we obtained in Question 1 and we already have the id_actor and actor_id correspondences. We could easily map the top 10 page ranked nodes by sorting the obtained pagerank vector. After sorting the pagerank vector, we illustrate the pagerank score distribution as following graph, and it is obvious that most actors/actresses have the score between 0 to 0.00008.

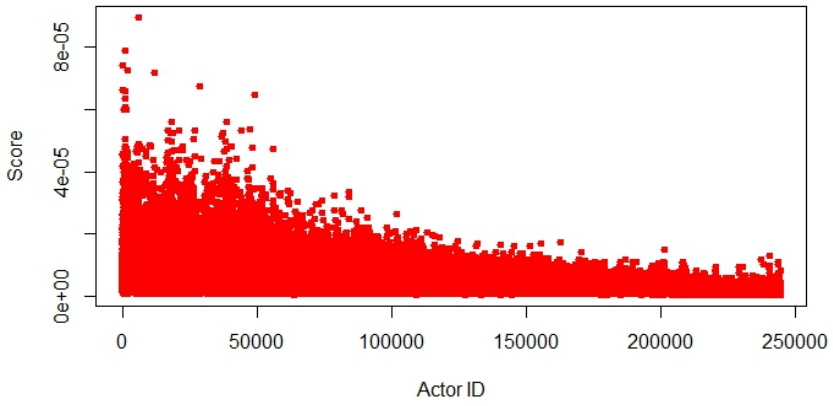

Figure 3.1. PageRank Score vs. Actor/Actress ID (Threshold = 5)

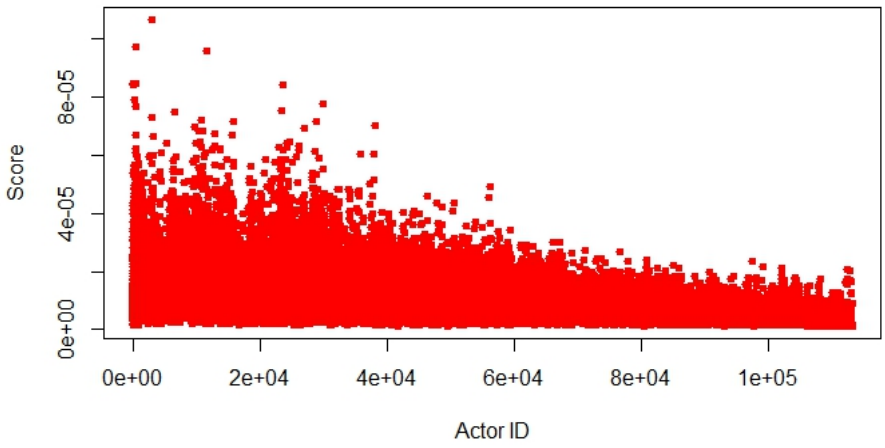

Figure 3.2. PageRank Score vs. Actor/Actress ID (Threshold = 10)

We run the pagerank algorithm with damping = 0.85 and algo = "prpack" on our weighted directed graph and we get the names of those in top 10 pageranks with the help of **actor_id.txt**. The names are listed in following Table 3.1.

Node ID	Actor/Actress Name	PageRank Score
117937	Roberts, Eric (I)	0.00008949914
140484	Trejo, Danny	0.00007896057
66948	Jeremy, Ron	0.00007401989
60911	Hitler, Adolf	0.00007245911
70569	Kaufman, Lloyd	0.00007171825
117130	Riehle, Richard	0.00006769812
65574	Jackson, Samuel L	0.00006629781
33602	De Niro, Robert	0.00006589041
85738	Madsen, Michael (I)	0.00006487603
32457	David, Keith (I)	0.00006357528

Table 3.1. Actors/actresses with top 10 PageRank scores (Threshold = 5)

Node ID	Actor/Actress Name	PageRank Score
56866	Roberts, Eric (I)	0.00010664830
67499	Trejo, Danny	0.00009758407
29215	Hitler, Adolf	0.00009611907
16135	De Niro, Robert	0.00008501890
31515	Jackson, Samuel L.	0.00008497271
56467	Riehle, Richard	0.00008440037
32130	Jeremy, Ron	0.00008439203
65947	Tatasciore, Fred	0.00007916428
15580	David, Keith (I)	0.00007913648
41288	Madsen, Michael (I)	0.00007766389

Table 3.2. Actors/actresses with top 10 PageRank scores (Threshold = 10)

Our teammates cannot recognize any of these actors/actresses, probably because of the generation gap. Thus, we list the names of famous stars in our opinion, which include very famous Hollywood stars and many of them get Oscar Awards. And then we calculate their corresponding pagerank. Their names and pageranks are listed in Table 3.2.

Node ID	Actor/Actress Name	PageRank Score
16878	Depp, Johnny	0.00005764250
27258	Hanks, Tom	0.00009758407
67477	Travolta, John (I)	0.00005764250
109050	Thurman, Uma	0.00003733683
55726	Reeves, Keanu	0.00003995785
15209	Damon, Matt	0.00005644564

3632	Bale, Christian	0.00004167300
96187	Marceau, Sophie	0.00002295892
102225	Portman, Natalie	0.00003318704
90962	Johansson, Scarlett	0.00003808811

Table 3.3. Actors/actresses in our opinion PageRank scores (Threshold = 10)

From the results of top 10 famous celebrities, we can surprisingly find out that most of them have relatively low pagerank scores, and the highest is Tom Hanks, but still not comparable with those top 10 pageranks. There are many reasons for this phenomenon that we are not familiar 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. Second, most of these actors are old and have done movies back in decades, so even after doing many movies, very few users would visit them now and hence their page ranks have dropped down. Furthermore, higher pagerank scores are not necessarily equal to good reputations. Those not famous actors tend to try in every kind of movie and act in many movies just as a supporting role. Thus, they are likely to have a closer relationship and connection to many other actors so that they have a high pagerank score.

Eventually, we find out the top 10 most significant pairs by convert our original directed graph into the undirected one with weight as the mean value of the directed edges. Then we sort the weight of all edges, and select the top 10 id pairs as our significant pairs. During the programming, we also use a simple Python program to create a id to actor dictionary to speed our matching between id and actor/actress name. The relevant code could be found at **generate_id_actor.py**.

First actor/actress in pair	Second actor/actress in pair
Williams, Sarah Anne	Katou, Emiri
Williams, Sarah Anne	Got, Yko
Williams, Sarah Anne	Yoshida, Seiko
Yuki, Aoi	Williams, Sarah Anne
Tsukui, Kysei	Katou, Emiri

Shimizu, Kazuki	Katou, Emiri
Kingetsu, Mami	Got, Yko
Yoshida, Seiko	Kenn
Yoshida, Seiko	Egawa, Daisuke
von Gomm, Peter	Katou, Emiri

Table 3.4. Top 10 significant pairs (Threshold = 10)

From the results of significant pairs, we can find out that there are many names appeared more than once like Williams, Sarah Anne. The common part is that these pairs may occur between different actors and actresses, but these actors all have relatively high weight with their connected edges, which can reflect that they have collaborated in most of their participating movies so that the movies are totally intersected. However, their names are still not that famous and they do not appear in the top pageranks as well. Actually, their page ranks are relatively low compared with our top pageranks. That is because, the pagerank will not simply count for the two vertices links directly and the total connectivity should not be overlooked.

Problem 4. Undirected Graph of Movies with more than 5 actors/ actresses

1) Pre-processing: Remove all movies with less than 5 actors/ actress

In this problem, we firstly remove all movies with less than 5 actors/ actresses. And then return a list of movies with more than 5 actors. actresses.

We have a hash table with movies and their number of actors/ actresses to achieve this. Examples are like this:

'The Killing Zone (1991)': 11,
'A Western Heroine (1911)': 5,
'Everybody, Dance (1964)': 10,
'From Working House to Mansion (1909)': 0,
'El observador (2003)': 0,
'My Hairy Cream Pie 2 (2006)': 0,
'Felix Wakes Up (1922)': 0,

'Dead (2013/III)': 0,
'A Texas Tale (2005)': 4,
'En passant (1943)': 0,
'Marginal Remarks (2010)': 0,
'Tobias and the Tree (2014)': 0,
"Prendeteci l'anima (2005)": 0,
'Scoundrel (2014/I)': 0,
'La mataviejitas (2006)': 0,
'OK Dude! (2007)': 0,
'Anal Torture (1995)': 0,
'Dream Date (2013/IV)': 1,
'Gefngnispostsack X4 Sdafrika (1982)': 0,

From the examples above, we find that there are some movies with 0 actors/ actresses because we remove the actors/ actresses with less than 5 movies. Therefore we do the similar process on movies and return a list of movies with more than 5 actors/ actresses. This process can prevent many isolated vertex in movie network which are unnecessary for community finding. After above process, we reduce the number of vertex from 1010947 to 246356.

2) Two mappings

After step 1, we construct two mappings using hash table. The first mapping is from movie to its actors/ actresses and the second mapping is from a actor/actress to the movies he/ she played in. In order to save space we use actor ID to represent actors/ actresses. Part of the examples are as below:

'Xin xi you ji (1987)':
['11672', '11688', '11696', '11874', '11978', '20454', '30171', '33351', '39218','39237', '39710', '40389', '52206', '65787', '67907', '70662', '89928', '94746', '96161', '112416'],

'Shin ohkubo monogatari (2013)':
['31396', '47045', '47578', '50759', '65554', '65599', '73187','88865', '92607', '98698', '100364'],

'Change of Heart (1993)':
['13664', '17831', '18148', '19928', '21129', '21270', '25846', '37356', '38064', '44708', '55351', '72300', '88489', '112798'],

'Vismayathumbathu (2004)':

['2752', '27244', '36618', '36648', '41261', '46033', '47196', '56749', '69288', '91387', '92936', '99465', '102135', '107978'],

The second mapping is from actors to the movies he/ she played in. We also use actor ID to represent the actor/ actress. The examples are as below:

'**27336**':
['A Fair Exchange (1916)',
'Busted Hearts (1916)',
'Chickens (1916)',
'Frenzied Finance (1916)',
'Furnished Rooms (1916)',
'In the Ranks (1916)',
'Love, Pepper and Sweets (1915)',
'Mixed and Fixed (1915)',
'Payment in Full (1916)',
'Pressing Business (1915)',
'Rushing Business (1916)',
'Speed Kings (1915)',
'Stranded (1916/II)',
'Strangled Harmony (1915)',
'Tangled Ties (1916)',
'The Frame-Up (1916)',
'The Great Safe Tangle (1916)',
'The Man Hunters (1916)',
'The Midnight Prowlers (1915)',
'The Pretenders (1916/II)',
'The Reward (1916)',
'The Rivals (1916)',
'The Try Out (1916)',
'Their Honeymoon (1916)',
'Their Wedding Day (1916)',
'This Way Out (1916)',
'Ups and Downs (1915)',
'Watch Your Watch (1916)'],

'**109512**':
['Camping Out (1928)',
'Choose Your Weapons (1922)',

'Dancing Man (1934)',
"Dante's Inferno (1935)",
'Daybreak (1931)',
"Everything's Ducky (1934)",
'Fish Feathers (1932)',
'Fugitive Lady (1934)',
'Gigolettes of Paris (1933)',
'High Hats and Low Brows (1932)',
'I Have Lived (1933)',
'Jitters the Butler (1932)',
"Let's Live Tonight (1935)",
'Love and Hisses (1934)',
'Man of the World (1931)',
'Manhattan Knights (1928)',
'Murder in Trinidad (1934)',
'No Children (1929)',
'No Picnic (1928)',
'No Sale Smitty (1928)',
'Paris Interlude (1934)',
"Princess O'Hara (1935)",
'Puckered Success (1929)',
'Symphony of Six Million (1932)',
'Ten Modern Commandments (1927)',
'The Back Page (1934)',
'The Captain Hates the Sea (1934)',
'The Comeback (1930)',
'The Lone Wolf Returns (1935)',
'The Midnight Adventure (1928)',
'Two Against the World (1932)',
"Uncle's Visit (1929)",
"Up in Mabel's Room (1926)",
'When Caesar Ran a Newspaper (1929)'],

These two hash tables will help us efficiently compute the jaccard index of the actor/ actresses sets between two movies as we can utilize 3 for loops to reduce the unnecessary computations.

3) Compute Jaccard Index

Jaccard index between 2 movie nodes, is the measurement of similarity between finite sample sets of actors/ actresses list of movie nodes, and is defined as the size of intersection divided by size of the union of the sample sets. The hashmap constructed before was used to find the weights of each movie with respect to all other movies. We calculated Jaccard Index by using following formula.

$$\text{Jaccard Index}(i, j) = \frac{\text{Intersection of list of actors/ actresses of movie}(i, j)}{\text{Union of list of actors/ actress of movie}(i, j)}$$

4) Construct movie network

We then used the Jaccard Index calculated above as weights to construct a movie network. Examples of the movie network are as below:

ahbap avuslar (1975)		Adile Teyze (1983)		0.030303030303030304
ahbap avuslar (1975)		Badem sekeri (1963)		0.058823529411764705
ahbap avuslar (1975)		Bir dag masali (1967)		0.029411764705882353
ahbap avuslar (1975)		Bizim Aile (1975)		0.03225806451612903
ahbap avuslar (1975)		Canim Kardesim (1973)		0.13793103448275862
ahbap avuslar (1975)		Dertli pinar (1943)		0.03225806451612903
ahbap avuslar (1975)		Dokunmayin Sabanima (1979)		0.06521739130434782
ahbap avuslar (1975)		Evet mi? Hayir mi? (1974)		0.02702702702702703
ahbap avuslar (1975)		Glen Gzler (1977)		0.029411764705882353
ahbap avuslar (1975)		Hababam Sinifi (1975)		0.03225806451612903
ahbap avuslar (1975)		Hababam Sinifi 3,5 (2006)		0.03333333333333333
ahbap avuslar		Hababam Sinifi Askerde		0.03333333333333333

(1975)		(2005)		
ahbap avuslar (1975)		Hababam Sinifi Merhaba (2004)		0.027777777777777776
ahbap avuslar (1975)		Hababam Sinifi Uyaniyor (1977)		0.03125
ahbap avuslar (1975)		Keriz (1985)		0.1
ahbap avuslar (1975)		Lks hayat (1976)		0.06060606060606061
ahbap avuslar (1975)		Oh olsun (1973)		0.03333333333333333

The first two columns represent two movies and the last column represents the weight between them.

Problem 5. Community finding and tagging on the movie network

1) Community Finding using fast greedy newman algorithm

In this problem, we first simplify the movie network generated in problem 4 by removing the duplicated edges. The duplicated edges are caused by double computing of the weights between any two movies.

After that, we do a community search using Fast Greedy Newman Algorithm for the movie network generated in problem 4. This process was time consuming and took us around 2 days to complete and we finally got 67 communities.

2) Tagging community with genres

After step 1, we tagged each community with genres that appear in 20% or more of the movies in the community. The community index and tagged genres are shown as below.

We observe the top genres of each community and find that the most frequent tag is “Drama”. It appears in 32 communities. And some of the second genres’ frequencies are also more than 20%, and these genres could also been used to tag a community. Therefore we can conclude that a single community are be tagged in one or two genres.

Index	Tagged Genre	Index	
1	Drama	35	Short
2	Drama	36	Drama
3	Drama	37	Short
4	Drama	38	Drama, Thriller
5	Drama	39	Drama
6	Drama	40	Drama
7	Drama	41	Comedy, Short
8	Drama	42	Short
9	Drama	43	Drama, Comedy
10	Drama	44	Comedy, Short
11	Drama	45	Drama, Short
12	Drama	46	Short
13	Drama	47	Drama
14	Drama	48	Short, Thriller
15	Drama	49	NA
16	Drama	50	Drama, Short
17	Drama	51	Drama
18	Drama	52	Documentary
19	Drama	53	Short
20	NA	54	Romance, Thriller
21	Drama	55	Crime
22	Comedy, Drama	56	Drama
23	NA	57	Short
24	Drama, Short	58	NA

25	Short	59	Short
26	NA	60	NA
27	Drama	61	Drama, Family, Musical
28	Comedy, Short	62	NA
29	Action	63	Comedy, Drama, Short, Thriller
30	NA	64	Comedy, Short
31	Drama	65	Romance, Short, Thriller
32	Comedy	66	Short
33	Comedy, Thriller	67	Mystery, Short
34	NA		

3) Tag information analysis

- Each of the tags gives us information of the most frequent types of the movies in the communities. The most frequent tag "Drama" means that the majority of the movie genres in the original data is "Drama". Also we can infer the similarity between two communities using the tags.
- Some of the communities may have no tag because not all movies with more than 5 actors/ actresses appear in the **movie_genre.txt** file. For simplicity, we only remain the communities with tags.

Problem 6. Adding 3 nodes into network and return top 5 nearest neighbors

1) Adding 3 nodes into network

In this problem, we first add 3 new nodes into network:

- Batman v Superman: Dawn of Justice (2016)
- Mission: Impossible - Rogue Nation (2015)
- Minions (2015)

2) Return top 5 nearest neighbors

After adding 3 nodes and the community search again, we have below results:

New Movie	Batman v Superman: Dawn of Justice (2016)	Community 14
Nearest Neighbors	Man of Steel (2013)	Community 14
	Lennon or McCartney (2014)	Community 14
	Beyond the Golden Age (2016)	Community 9
	Iron Man 3 (2013)	Community 14
	Star Wars: The Old Republic (2011)	Community 13

New Movie	Mission: Impossible - Rogue Nation (2015)	Community 12
Nearest Neighbors	Broadway: Beyond the Golden Age (2016)	Community 12
	Ted 2 (2015)	Community 14
	The Dark Knight Rises (2012)	Community 12
	The Dark Knight (2008)	Community 14
	Star Trek (2009)	Community 12

New Movie	Minions (Voice) (2015)	Community 9
Nearest Neighbors	The Lorax (2012)	Community 9
	Inside Out (2015)	Community 9
	Despicable Me 2 (2013)	Community 17
	Up (2009)	Community 9
	Surfs Up (2007)	Community 9

We now find the top 5 neighbors of each of these 3 movies and the community to which they belong to. Neighbors are found based on the the similarity of Jaccard Index. We can find that each of the movies and its neighbors are belong to the same community.

Problem 7. Predict the ratings of the above 3 movies

We predicted the 3 movies based on the community and movie_rating.txt. Assume that movies with similar actors and similar styles have the approximate equal rating.

By question 6, we find the nearest neighbors of these 3 movies, as well as their communities. For each of the added movie, find the ratings of its 20 nearest neighbors and calculate the average of ratings of these movies. This average rating is the predicted rating of the added movies. The results are showed by the following table, compared to the actual IMDb rating:

Movie	Predicted Rating	Actual Rating
Batman v Superman: Dawn of Justice(2016)	6.5	6.7
Mission: Impossible- Rogue Nation(2015)	7.0	7.4
Minions(2015)	7.2	6.4

Table 7.1 Predicted Ratings of the 3 Movies

There is some discrepancy between the predicted rating and the actual rating. It is because the amount of the nearest neighbors is too small (neighbor=20), and we don't consider the other

movies in the same community. Naturally, we came up with the idea that the nearest neighbor and other movies in the same community weigh 50%, respectively. Then we get the modified predicted rating, and showed by the following table:

Movie	Modified Rating	Actual Rating
Batman v Superman: Dawn of Justice(2016)	6.2	6.7
Mission: Impossible- Rogue Nation(2015)	6.3	7.4
Minions(2015)	6.5	6.4

Table 7.2 Modified Predicted Ratings of the 3 Movies

However, the error still exists when using the modified rating. We think the error may come with several reasons:

1. Different levels of investment. Two movies can have similar actors, but different costs related to the rent, decorations and post-process techniques.
2. Different time. Decades of time span can have a great impact on people's taste towards movies.
3. Different director. Good director can produce movies with very high ratings with limited investment and actors who are not very famous.

Consider the reasons mentioned above, it is natural that our predicted movie rating is different from the actual IMDb movie rating. However, if we get more data of the movie and consider them all into our rating model, the results will be much better.

Problem 8. Train a regression model and predict the ratings of the above 3 movies

In this question, we try to use two different models to predict the ratings of these 3 movies. The first model is linear regression model, and the second is random forest regression model.

Linear regression model is quite simple and familiar with. Using the data of 5 PageRank of the actors (with 5 floating point values) in each movie, and the top 100 directors, we have 105 inputs in total. Using these inputs, we can train the model to produce a linear function of these 105

inputs. Then it is very easily to predict the rating of these added movies. The target of the linear regression model is to minimize the error, which is measured with variance, or RMS value
.
The result is showed as the following table,and the RMS error is the average value in all of the experiments:

Predicted Ratings of the 3 Movies with Linear Regression Model			
Movie	Predicted Rating	Actual Rating	RMS error
Batman v Superman: Dawn of Justice(2016)	6.1	6.7	1.26-1.29
Mission: Impossible- Rogue Nation(2015)	6.2	7.4	
Minions(2015)	6.2	6.4	

Table 8.1 Predicted Ratings of the 3 Movies with Linear Regression Model

The linear regression model is quite simple, and works well in most of the times. However, consider that some function is not linearly related, we consider to use another model, random forest regression model, to predict the rating of these 3 movies.

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1, ..., x_n$ with responses $Y = y_1, ..., y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:
For b = 1, ..., B:

1. Sample, with replacement, B training examples from X, Y; call these X_b, Y_b.
2. Train a decision or regression tree f_b on X_b, Y_b.

In this question, the input x is the same us linear regression model, and the responses y is the other movies with the ratings that we already know. If we keep adding movies to train this random forests regression model, the result will become more precise to the actual rating.

After the training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

And this is the predicted rating each of the added 3 movies. Also, the target of the random forest regression model is the same us linear regression model, to minimize the RMS error.

The result is showed as the following table, and the RMS error is the average value in all of the experiments:

Predicted Ratings of the 3 Movies with Random Forest Model			
Movie	Predicted Rating	Actual Rating	RMS error
Batman v Superman: Dawn of Justice(2016)	6.1	6.7	1.25-1.28
Mission: Impossible- Rogue Nation(2015)	6.1	7.4	
Minions(2015)	6.1	6.4	

Table 8.2 Predicted Ratings of the 3 Movies with Random Forest Model

From the results of these two models, we surprisingly found that the result is quite similar, and the ratings of each movie is quite close. We think that is because the input of the model is not enough to predict the rating of these movies. As stated in problem 7, there are many factors can influence the rating of a movie, which is not concluded in this project. The other thing we must consider is that whether the PageRank score and top 100 director have causal effect on the rating of a movie, or whether it is just correlation effect. Anyway, the rating is determined by the customers.

Problem 9. Predicting the ratings with a bipartite graph

In this question, we try to construct a bipartite graph to predict the ratings of these 3 movies. A bipartite graph is a graph whose vertices can be divided into two disjoint sets U and V (that is, U and V are each independent sets) such that every edge connects a vertex in U to one in V. To be specific, an actor or actress is connected to all the movies that he or she participated in.

The algorithm is as follows:

1. A rating of an actor is the average rating of the movie he played in.

   If an actor a_1 played movie which rates $r_1, r_2 \ldots r_n$, respectively, then the rating of an actor is defined as:

$a_1 = (r_1 + r_2 + \ldots + r_n)/n$

2. A rating of a movie m_1 is the average rating of the actors that played in the movie. If the movie is played by actors which rates a_1, $a2 \ldots a_n$, respectively, then the rating of a movie is defined as:

$m_1 = (a_1 + a_2 + \ldots + a_n)/n$

And this is a quite straightforward linear model.

The result is showed as the following table:

Movie	Predicted Rating	Actual Rating
Batman v Superman: Dawn of Justice(2016)	6.4	6.7
Mission: Impossible- Rogue Nation(2015)	6.5	7.4
Minions(2015)	6.9	6.4

Table 9.1 Predicted Ratings of the 3 Movies with Bipartite Graph

The ratings with Bipartite Graph seems to be better than all the predictions in question 7 and 8. This is because actors and actresses play an important role in the movies, and some customers and fans are attracted only by them.

However, there is some limitations in this rating scheme:

1. Actors and actresses in a movie should weigh differently, not with the same weight. This is very obvious, since that the most famous one can play a much more important role than several new and unfamiliar actors and actresses. Consider that if an actor played in many movies, he must be very familiar and has a relative high rating compared to other actors. Otherwise, say that an actor is very poor in acting, then the movie company will not let him play in more movies. This survivorship bias, called in statistics, will underestimate the ratings of the movie.
2. The rating of an actor or an actress should be somewhat different, especially for a famous one. For example, when an actor played his first movie, his rating maybe quite low. However, as time passes, and he became famous, his rating can be quite high. So it is necessary for us to consider the time span.
3. As stated in question 7 and 8, there are some other factors that can impact on the ratings, which are not included in this question. The factors include investment, people's taste, and directors.

3. Useful Links

[1] https://stackoverflow.com/questions/509211/explain-pythons-slice-notation
[2]https://stackoverflow.com/questions/11108416/trouble-understanding-count-multiple-and-simplify-in-igraph
[3] Our intermediate data files: https://drive.google.com/drive/folders/0B2I9Ux-1OJOmR0F3VW9QOE5SZmc?usp=sharing
[4] Initial files: https://ucla.app.box.com/s/9zc9wlhecqxja7y3kmfpxxxfoz2od5be