EE232E Project 2 Report

TEAM MEMBERS: Qinxin Li 704774783

Jiatong Chen 404361029

Yuliang Fang 404759010

In this project, we used a graph of IMDb movie networks. In the first part of the project, we created movie network based on actors and actresses to find out the rankings of the actors and actresses using PageRank algorithm. We also created network of movies to find the community structure in the network. In the second part of the project, we use multiple methods to predict the ratings of movies based on cluster neighbors or featured actors, and we compare among those predicted ratings.

1.Preprocessing data

In this exercise we use the following files to create network.

- actor_movies.txt
- actress_movies.txt

The lines in the file contains the actor's name in the first column and his movies follows. The movie names includes tags like "(voice)","(uncredited)" or others. Hence we remove all the tags from the movie name string. The objective of this part is to create a network of actors and actress who acted in more than 10 movies. We use Spark to process the textdata. First, we use sc.textfile and map to split the lines. Then we use flatMap function to generate (name,movie) tuples. Later, we convert the rdd to dataframe object, group the dataframe on actor's names and filter the actors who act in movies less than 10.

2.Designing actors/actress network

In this part, we create a directed weighted graph G(V,E) based on the data from Part 1. The parameters has the following meaning:

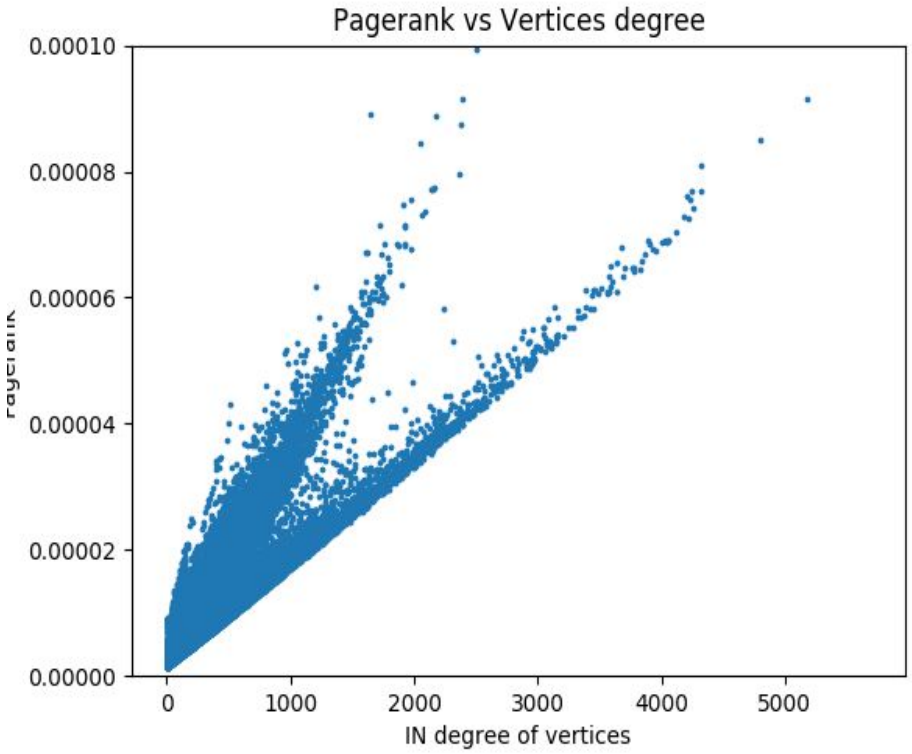
- Vertices(V) are actors and actress.
- Si = {m|i ∈ V, m is a movie in which i has acted}, i.e. set of movies an actor/actresses has acted.
- E = {(i, j)|i, j ∈ V, Si ∩ Sj != φ}, i.e. an edge is created if two actors (vertices) have at least one common movie.
- Weight (W) of each edge (i to j) was assigned as W = |Si ∩ Sj|/ |Si|, i.e. ratio of the movies acted together by actors/actresses i and j to the

movies acted by actor/actress i. Hence the graph is directed as each edge has different weight.

As the network is directed, we must create a dataframe to contain the edgelist, in which the first column is the source vertex, the second column the target vertex and the third the edge weight. Therefore we first group the (name,movie) pairs on movie names and aggregate the names to list. Then we use flatmap and list.combination method to create (name1,name2) pairs from each list of names. Finally we use reduceByKey() method to count the (name1,name2) pairs. Since we created a name_count dataframe before, we can divide the count of (name1,name2) pairs by each count corresponding to name1 and name2 to get the edge weight of directed edges. We use scala to write the code, which has a powerful function of mapreduce method. It is estimated that if we use iterations to count all (name1,name2) pairs, the time cost would be 1 hour or more, while our spark code only needs 1 or 2 minutes. We write it to files and use igraph.read() method to read the graph.

3. Pagerank on actors/actress network

PageRank is a link analysis algorithm and it assigns a numerical weighting to each element of a network and its mostly used to determine the importance of the node in the network. The algorithm returns the probability distribution representing the likelihood of a random walker visiting a particular node in the network.
We ran the pagerank algorithm on the network created in above exercise. The pagerank score of node against node degree is plotted below:



We can see that the vertices with high in-degree has high pagerank score, which means that the most-often visited vertices has high pagerank score. The plot has 2 linear correlations, which means that the pagerank is proportional to the degree of the node. Intuitively in the created network, the actor/actress who have acted with different actor/actress has high PageRank score as their node degree will be high. The table below has the top 10 actors/actresses in the network defined by the PageRank score.

Name	PageRank
Aoyama, Minami (I)	0.0001296281243420695
Aavik, Evald	0.00010844655202176335
Abhilasha (I)	0.00010764254768526949
Adjadj, Marc	0.00010195647533007299
Anderson, Roy (II)	9.935341453265745e-05
Adonis, Mae-ann	9.158199953362323e-05

French, Samuel	9.148281374356819e-05
Baxter, Anne (I)	8.89976695279989e-05
Arena, Giancarlo	8.872785732115076e-05
Aav, T\xef\xbf\xbdnu	8.737786369327185e-05

We can see that we are not familiar with those actors with high pagerank.
We also get the pagerank ofsome celibrities:

Johansson, Scarlett	1.06890157789e-05
Hanks, Tom	7.94635991574e-06
Pitt, Brad	3.18019019398e-06
Portman, Natalie	7.11046340887e-06

Their pagerank score is not so high, though. It is probably due to that those actor with high pagerank scores plays a small role in many films, thus they have many more cooperation times, which means they have higher degree than others.

4. Construct Movie Network

(TA has revised the question so that we remove all movies with less than 10 actors/actresses on list)

Here we use **PySpark** RDD and DataFrame to process this large data set. Only within 15 minutes is needed to construct the movie network on a single Mac. The whole process is as follows,

i) Read actor_movies.txt and actress_movies.txt files into RDD data structure and filter out actors and actresses with less than 10 movies. Then we use flatMap operation on all movies corresponding to each name, obtaining (one name, one movie) objects.

ii) Transform the RDD objects into DataFrame, then group by “movie” and obtain a new DataFrame with (one movie, list of names) Row objects. Then we filter out movies with less than 10 actors and actresses. Then we count the number of actors/actresses for each movie and store the data as (movie, count) table.

iii) Transform the DataFrame with (one movie, list of names) Row objects back to RDD and use flatMap to get (one movie, one name) objects. Then again we groupby “name” and obtain a new DataFrame with (one name, list of movies) Row objects. Then we use flatMap operation and *itertools.combinations(movie,2)* list of movies to get (one movie1, one movie2, 1) objects. After this, we use *reduceByKey(lambda a,b : a+b)* to get (one movie1, one movie2, # of common actors/actresses) table.

iv) Join (one movie1, one movie2, # of common actors/actresses) table with two (movie, count) tables (in step ii)) on keywords *movie1==movie* and *movie2==movie.* Then we can easily calculate the Jaccard Index (weight) of movie1 and movie2 from,

$$Jaccard\ Index\ =\ \frac{\#\ of\ common\ actors/actresses}{count1+count2-\#\ of\ common\ actors/actresses}$$

The the table (one movie1, one movie2, Jaccard Index) serves as input for python-igraph to generate a graph.

Our network has 98749 nodes and 29766038 edges in the end.

5. Community finding on Movie Network

(The communities generation is using **python-igraph package**. All data manipulations are using **Pandas DataFrame**)

Here we use Fast Greedy Newman algorithm to find community structure of the undirected movie network generated from Question 4. The algorithm takes almost 4 hours to find 24 communities in the movie network. Then based on the genres of all movies present in a community, we tag the community with genres that show up in more than 20% (threshold) of the movies in this community. Here the keyword “unknown” appear in the Genre column because some of movies in the network does not show up in movie_genre.txt file. We set their genre to be unknown

Community Membership	**Community Size**	**Genre**
1	18293	

2	4636	Drama, unknown
3	3689	Drama
4	2912	Drama
5	1196	Drama, Romance
6	15966	Comedy, Drama
7	24781	Thriller
8	1183	Comedy, Drama
9	5204	Drama
10	5655	Drama, unknown
11	3725	Romance, unknown
12	874	Comedy, Drama
13	2696	Comedy, Drama
14	3639	Drama
15	1901	Drama
16	1075	Drama
17	340	Short
18	58	Drama, Romance
19	781	Comedy, Drama
20	6	Drama, Thriller
21	38	Drama, unknown
22	27	Drama, War
23	72	unknown
24	2	Drama, Romance

If we set the threshold to be 15%, then the table appear to have more genres for each community.

Community Membership	**Community Size**	**Genre**

1	18293	Drama, Romance, Western
2	4636	Drama, unknown
3	3689	Drama
4	2912	Action, Comedy, Drama
5	1196	Drama, Romance
6	15966	Comedy, Drama
7	24781	Drama, Thriller
8	1183	Comedy, Drama, Romance
9	5204	Comedy, Drama, unknown
10	5655	Comedy, Drama, unknown
11	3725	Drama, Romance, unknown
12	874	Comedy, Drama
13	2696	Comedy, Drama
14	3639	Drama
15	1901	Drama, Romance
16	1075	Drama, War
17	340	Short
18	58	Comedy, Drama, Romance
19	781	Comedy, Drama
20	6	Drama, Romance, Short, Thriller
21	38	Comedy, Drama, unknown
22	27	Drama, War
23	72	Drama, unknown
24	2	Drama, Romance

6. Return the top 5 nearest neighbors for 3 movies:

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

The rank of the neighbors of the given movie is determined by the weight of edges connecting the given movie to its neighbors. The larger the weight, the higher the rank. Hence we find the top 5 neighbors of 3 given movies and tabulate them as below.

Neighbors of *Batman v Superman: Dawn of Justice (2016)* :

The community membership of *Batman v Superman: Dawn of Justice (2016)* is **6**		
Top 5 Nearest Movie	**Community membership**	**Movie Genre**
Eloise (2015)	6	Thriller
Man of Steel (2013)	6	Sci-Fi
Demoted (2011)	6	Comedy
Love and Honor (2013)	6	War
Real Steel (2011)	6	Sport

Neighbors of *Mission: Impossible - Rogue Nation (2015)* :

The community membership of *Mission: Impossible - Rogue Nation (2015)* is **6**		
Top 5 Nearest Movie	**Community membership**	**Movie Genre**
Muppets Most Wanted (2014)	6	Musical
Fan (2015)	6	Drama
Now You See Me: The Second Act (2016)	6	Thriller
Phantom (2015)	6	Thriller
Patient Zero (2015)	6	Horror

Neighbors of *Minions (2015)* :

The community membership of *Minions (2015)* is **6**		
Top 5 Nearest Movie	**Community membership**	**Movie Genre**
Inside Out (2015)	6	Family
The Lorax (2012)	6	Fantasy
Gake no ue no Ponyo (2008)	6	Fantasy
Surf's Up (2007)	6	Sport
Up (2009)	6	Adventure

We notice that the given movie share the same community membership with their top 5 neighbors. It means that those movies that share most common actors and actresses usually fall into the same community. However, as we can see from the Movie Genre column, more common actors and actresses does not necessarily yield more common movie genres, which is reasonable.

For Minions (2015), its top 5 neighbors are animated. Therefore, the same set of supporting actors and actresses might have intention to work at several animated movies together.

7. Ratings Predictions

In this problem, we use the movie_rating.txt to predict the ratings of three movies: Batman v Superman: Dawn of Justice (2016), Mission: Impossible - Rogue Nation (2015) and Minions (2015) by using the movie network.
Since the data is very large, we did the pre-processing as the above problems. We remove the actors with less than 5 movies. By doing this, we can map each movie to its corresponding ratings.

We do the ratings prediction by two ways:
a. Movies in the same community: Average ratings of all the nodes in the community which the predicting nodes exists.
For this method, we first take the community the predicted node belongs to, then, we take all the nodes' ratings in that community and take the mean average as the predicted ratings.
b. Based on neighborhood movies: Average ratings of all the nodes which are neighbors to the predicting node.
Same as the above method, we take the ratings of the nodes' neighbors and take the mean average of these ratings as the predicted ratings.

After doing the two methods, we can take the average of them. After that, we can compare the predicted ratings with the actual IMDb ratings. The actual ratings are: 7.1, 7.5, 6.4.

The results are shown below:

Ratings	Batman v Superman: Dawn of Justice (2016)	Mission: Impossible - Rogue Nation (2015)	Minions (2015)
community	**6.07**	**6.13**	**6.20**
neighborhood	**6.11**	**6.00**	**6.40**
average	**6.09**	**6.07**	**6.30**
actual	**7.1**	**7.5**	**6.4**

Analysis of results:
We can see that the movie- Minions (2015) has a closer predicted rating to its actual rating. The other two have obvious difference between the prediction and reality. This may because of that the movies in the community or the neighbor movies are not that related to the target movie, so that the reference ratings cannot present much about the target movie.
From the result, it can be shown that this prediction is not that reliable sometimes, since the it depends on the genre of the movies in the community or the neighborhood.

8. Use additional features and do Ratings Predictions

In this problem, we also only consider the movies with more than five actors.

The additional features are:
a. top 5 page-ranks of the actors (five floating point values) in each movie.
For this part, we take the top 5 page-ranks of actors who acts in a particular movie were assigned to the first 5 dimensions of the feature vector.
b. If the director is one of the top 100 directors or not (101booleanvalues).
Using the director_movies.txt and the movies_rating.txt, we checked if each director is in the top 100 director or not. First we obtain the director's name of the particular movie, and we create a 100 dimension boolean value vector with all 0 values, the

100 positions correspond to the top 100 movies, from the top1-top100. If the director directed a movie in top 100 movies, then we place a 1 in the corresponding rank place into the vector. If the director does not have any movies in the top 100 movies, then we keep the 0 at that position in the vector.

For example, if we are currently at movie A, and its director is B. Then we check how many movies the director directed are in top 100 movies. If, we say, the director B of movie A has 2 movies in top 100, which are in rank 10 and 20. Then we place 1s at position 10 and 20 in the boolean vector, and the feature vector will be the top 5 page-rank actors in movie A.

For this problem, we randomly picked 5 movies to train the model. We use the code *np.hstack()* and *np.vstack()* in python to get the final feature matrix.
After creating the matrix, we train a linear regression model.

The summary for our regression model is:
Observations: 194882
Df Residuals: 194817
Df Model: 65
R squared: 0.958
Adjusted R squared: 0.958
F-statistic: 6.526e+04

Our R-squared value is very close to 1, which means our model explains the variance of the response data around its mean pretty well.

Now, we extract the features for the 3 movies: Batman v Superman: Dawn of Justice (2016), Mission: Impossible - Rogue Nation (2015) and Minions (2015) and get the predicted ratings.

The predicted ratings are shown below:

Ratings	Batman v Superman: Dawn of Justice (2016)	Mission: Impossible - Rogue Nation (2015)	Minions (2015)
Predicted ratings	**6.19**	**6.17**	**6.18**
Actual ratings	**7.1**	**7.5**	**6.4**

Analysis of results:

We can see from the result that the predicted ratings are very similar and the first two predictions are not that close to the actual value. The summary of the model, however, shows our regression model is not bad.
We think the problem here is that the features are actually not good enough. Especially the top 5 page-rank actors feature. For example, some actors who acts a lot in many movies have higher page-ranks, but this does not mean that they are the main actors in that movie. Usually, most of the ratings of the movies depends on the lead actors, or, we say, the stars. So the page-ranks cannot be a perfect feature for ratings prediction.

9. (Bonus) Bipartite graph

In the mathematical field of graph theory, a bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint sets and (that is, and are each independent sets) such that every edge connects a vertex in to one in. Vertex sets and are usually called the parts of the graph. In this problem, the two vertex sets are actor sets and movie sets.

Our method for prediction is:
First, we need to assign a score to each actor. Suppose we have actor A. He acts in movies a1, a2……, an, n movies. The ratings of these movies are: R1, R2,……, Rn. So we have the actor's score: $(\sum_{i=1}^{n} Ri)/n$. Then, we need to decide the ratings of the movies. Suppose we have a movie: α, and there are n actors who acts in this movie. Since we already have the scores of each actor, we can do the prediction of ratings by: $(\sum_{i=1}^{n} Si)/n$, where Si are the scores of each actors.

We use 3 metrics to assign a score to an actor: the highest rating of his movies, the average rating of his movies and the top3 ratings of his movies. Then we predict the movie's ratings based on the average scores of the actors participating in this film.

Ratings	Batman v Superman: Dawn of Justice (2016)	Mission: Impossible - Rogue Nation (2015)	Minions (2015)
Highest Predicted ratings	**6.6**	**6.3**	**6.7**

Avg Predicted ratings	**5.8**	**5.4**	**6.0**
Top 3 Predicted ratings	**6.0**	**5.8**	**6.4**
Actual ratings	**7.1**	**7.5**	**6.4**

Analysis of results:
We can see that the Highest predicted ratings are higher than the other 2 predicted ratings and are closer to the actual ratings. It means that the rating of movies are influenced especially by the highest score of the actors.
We can see that the result of this method is better than the above two. It is because this method relates actors and movies together. This method helps eliminate irrelevant information. We have analyzed the problems for the above two methods. Both of them are affected by irrelevant information. The first one is because of the unrelated movie genre, and the second is because of the lacking of relationship between the actors and movies. However, for this method, both of the problems are solved by relating the actors and movies together.