EE232E: Graphs and Network Flows
Project 2 - IMDb Database Exploration

Mansee Jadhav - 204567818
Ronak Sumbaly - 604591897

June 7, 2016

Introduction

In this final project, we create and explore a network from the `IMDb movie dataset`. The data-set comprises of information like `actor/actress names` and the movies they have enacted in along with `movie ratings, directors and genres of movies` as well.

Question 1

Downloading & Pre-processing Data

The IMDb dataset files have 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, data.table`.

Since all actors don't have the same number of movies, reading the dataset in form of a table was not possible. But in order to enable us to do faster parsing we read the data in form of a table (delimited by : "\t\t") with pre-defined 6 columns. Since there are actors with less than 5 movies, columns are filled with `NA values` if no data is present. Since only 6 columns are read we are able to read the data at a faster pace. The rows which have `NA values` are simply removed giving us actors with > 5 movies.

Parsing Data

After obtaining all the actors/actress with > 5 movies we perform a following steps for parsing. Each of the step is performed using the `foreach` library which facilitates `%dopar%`. The steps included,

1. Initial encoding of all Latin and non-ASCII characters to empty.
2. Trim all the leading and ending blank spaces.
3. Removing all the extravagant information following a movie-name e.g {{SUSPENDED}}, (voice)

Statistics of the Data

$$\texttt{Total number of actors \& actresses with > 5 movies} = 244290$$
$$\texttt{Total number of movies (adding movies of all actors \& actresses)} = 4024168$$
$$\texttt{Total number of unique movies} = 531444$$

Question 2

Construction of Weighted Directed Graph

After obtaining the pre-processed data, we created a `hashmap of movie` $\rightarrow$ `All Actors/Actress`. In order to accomplish this we indexed all the unique movies. Correspondingly each actor and actress were mapped to an numeric index value. We then created `hashtable` mapping actor/actress indexes {values} list corresponding to a movie {key} by binding all movies with their actors using `data.frame`.

Hashtable Representation

Movie Name	Actor / Actress Index
Movie 1	$Actor_1$, $Actor_2$, $Actress_1$, $Actress_2$, $Actress_3$, $Actress_4$
$\vdots$	$\vdots$
Movie n	$Actor_3$, $Actor_2$, $Actor_5$, $Actor_7$, $Actress_2$

In order to create the weighted directed graph the following steps were followed (written in Python),

1. Using the generated `hashtable`, we formed pair of actors belonging to each movie.
2. We collected all these generated pairs, and counted the number of times each pair was repeated. Since actor/actress repeats in movies, pairs of actor/actress would repeat throughout the hashtable. This count gave us the `number of movies that the pair share`.
3. All the unique generated pairs of (i, j) were considered as edges within the graph, and their corresponding weight was calculated using the number of movies common to i and j calculated from step 2, divided by number of belonging to i (computed using the first hashmap).

Weighted Directed Graph Details

$$\texttt{Number of edges within the graph} = 57846322$$
$$\texttt{Number of vertices} = 244290$$

Question 3

Page Rank Algorithm on Actor/Actress network

After constructing the `IMDb network` above we calculated the page rank of all the node using `page.rank` of `iGraph package`. Since the vertices are numbered in the same order as the list of actors/actress we obtained in `Question 1` we were easily able to map the top 10 page ranked nodes by sorting the obtained page rank vector.

Top 10 Page Ranks

The top 10 Page Ranks vertices obtained from the page rank algorithm are provided below.

Actor/Actress Names	Page Rank	Number of Movies
Roberts, Eric (I)	8.9455e-05	298
Jeremy, Ron	8.5272e-05	637
Trejo, Danny	7.8565e-05	241
Tatasciore, Fred	7.6064e-05	355
Kaufman, Lloyd	7.3294e-05	301
Hitler, Adolf	7.0732e-05	379
Jackson, Samuel L.	7.0226e-05	159
Flowers, Bess	6.8052e-05	828
Riehle, Richard	6.734e-05	197
David, Keith (I)	6.4854e-05	178

Table 1: Top 10 Page Rank Actor/Actress

Apart from `Samuel L. Jackson and Adolf Hilter` most of the other names are of those actors/actress that our generation isn't familiar with, hence not famous to us. A page rank of a page is defined by the number of other important pages pointing to it and this is evident by the following vertex information.

Consider, `Roberts, Eric (I)`, one of Hollywood's edgier, character running around and about for decades. He is the son of Betty Lou (Bredemus) and Walter Grady Roberts, who are themselves one-time actors and playwrights. His siblings are also actors Lisa Roberts Gillian and Julia Roberts (familiar name). Naturally the actor shares connections with many other actors and thus has a higher page rank.

Most of them are `nominated for Oscars, hence they are a topic of discussion in social media, thus gaining larger page importance.`

Top 10 Famous Actors/Actress

In our opinion, top 10 famous actors/actresses would be the one with largest number of movies. Thus when we fetch page ranks for such famous entities, we get below results:

`Actor/Actress Names`	`Page Rank`	`Number of Movies`
Blanc, Mel	3.5909e-05	1065
Brahmanandam	4.3462e-05	973
Onoe, Matsunosuke	1.4217e-05	927
Flowers, Bess	4.3235e-05	828
Hack, Herman	6.8052e-05	668
Phelps, Lee (I)	8.5272e-05	647
Jeremy, Ron	3.1324e-05	637
Cobb, Edmund	2.2458e-05	633
O'Connor, Frank (I)	3.1170e-05	623
Kapoor, Shakti	4.2351e-06	618

Table 2: Top 10 Famous Page Rank Actor/Actress

Well - known actors do not show up in the high pagerank list

- One of the reason for these well known actors to not show up in high page rank lists is, most of these actors are old and have done movies back in decades. Thus even after doing many movies, very few user's visit them now and hence their page ranks have dropped down.
- Secondly, most of the famous actors have their independent pages within IMDb, hence they do not share linkage with pages of other actors and subsequently getting a lower page rank.

Question 4

Undirected Graph of movies with > than 5 actors/actress

For this question, we were asked to construct an undirected network of movie, considering only those movies with more than or equal to 5 actors/actresses. Weights were assigned as the `jaccard index`of the actors sets of 2 movies.

`Jaccard coefficient` between 2 movie nodes, is the measure of similarity between finite sample sets of actors/actresses list of movie nodes, and is defined as the size of the intersection divided by the 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 the other movies. We calculated Jaccard Index in `Python`, by using the following formula,

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

Undirected Movie Graph Details

$$\texttt{Number of edges within the graph} = 70640020$$
$$\texttt{Number of vertices} = 246378$$

Question 5

Community Structure using Fast Greedy Approach

On the built movie network, we applied the `Fast Greedy Newman Community Forming Algorithm`. No optimization was possible for this question and since the network was too large it took 19 hours to detect all the communities within the network. In the end we found `67 communities.`

Tagging Communities with Genre

Each movie in the network was assigned its corresponding `Genre` and using this information and the `community membership` obtained we were able to tag community with genre like `Drama, Thriller, Romance, Comedy` etc. if 20 % or more corresponding genre movies belong in the community. Step wise details are provided below,

- In order to accomplish this `movie_genre.txt` file was read in, cleaned and processed as in Question 1. We removed all the Latin Encoding, extra spaces and all the extravagant text following a movie name to facilitate matching.
- These movie genres were then assigned to their respective nodes within the movie network as the `genre attribute` to each network.
- Once all the nodes were assigned to their respective genre we looped across all the communities, to get their different genres count, then tagged that community with he genre that appear in 20% or more of the movies in the community.

Inference from tags

- Each of this tags, gives us information of type of movies that appear in communities. We observe that most tagged genres are "Drama". It is because the majority of the movie genres in the original data is "Drama". Also this tags can be used to understand the similarity between two or more communities.
- Some of the communities have a `No Tag` as their tag.genre because not all the movies have tag present in the `movie_genre.txt` file, hence for such movies we have replaced them with a "No Tag" genre.

Below is the information of the nodes within each community and their tag names:

Community Number	Frequency	Tag Genre	Community Number	Frequency	Tag Genre
1	5,626	Drama	34	193	No Tag
2	13,624	Drama	35	10	Short
3	16,851	Drama	36	473	Drama
4	57,548	Drama	37	23	Short
5	5,344	Drama	38	7	Drama, Thriller
6	45,773	Drama	39	6	Drama
7	6,309	Drama	40	5	Drama
8	44	Drama	41	7	Comedy, Short
9	14,848	Drama	42	6	Short
10	7,236	Drama	43	8	Drama, Comedy
11	1,902	Drama	44	8	Comedy, Short
12	8,386	Drama	45	12	Short
13	1,580	Drama	46	8	Drama, Short
14	37,274	Drama	47	8	Drama
15	2,487	Drama	48	12	Short, Thriller
16	1,420	Drama	49	10	No Tag
17	7,931	Drama	50	7	Drama, Short
18	1,094	Drama	51	6	Drama
19	3,979	Drama	52	7	Documentary
20	2,395	Drama	53	9	Short
21	1,343	No Tag	54	8	Romance, Thriller
22	879	Drama	55	7	Crime
23	14	Comedy, Drama	56	7	Drama
24	877	No Tag	57	5	Short
25	8	Drama, Short	58	5	No Tag
26	6	No Tag	59	5	Short
27	139	Drama	60	5	No Tag
28	20	Comedy, Short	61	4	Drama, Family, Musical
29	6	Action	62	5	No Tag
30	509	No Tag	63	4	Comedy, Drama, Short, Thriller
31	24	Drama	64	6	Comedy, Short
32	6	Comedy	65	3	Romance, Short, Thriller
33	2	Comedy, Thriller	66	3	Short
-	-	-	67	2	Mystery, Short

Question 6

Adding additional 3 nodes to the network

We now add additional 3 movie nodes into the network.

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

Top 5 nearest neighbors and their communities

We now find the top 5 neighbors of each of these movies and the communities to which they belong to. Communities for each of the above movies were found using the `community$membership`. Neighbors were found based on similarity of the Jaccard weights.

We can see that most of the found neighbors, belong to the same community as that of the main movie whose neighbors are found. These neighbors are retrieved based on the similarity between movies. Greater the similarity, more are the changes of them belonging to same community, thus lowering their distance between each other. The results obtained for each of the movies is given below,

Batman v Superman: Dawn of Justice (2016)

Batman v Superman: Dawn of Justice (2016) - Community 14		
KNN	Movie Name	Community
1	Ted 2 (2015)	14
2	Lennon or McCartney (2014)	14
3	Beyond the Golden Age (2016)	9
4	Star Wars: The Old Republic (2011)	13
5	Iron Man 3 (2013)	14

Mission: Impossible - Rogue Nation (2015)

Mission: Impossible - Rogue Nation (2015) - Community 12		
KNN	Movie Name	Community
1	Ted 2 (2015)	14
2	Broadway: Beyond the Golden Age (2016)	12
3	The Dark Knight Rises (2012)	12
4	Star Trek (2009)	12
5	The Dark Knight (2008)	14

Minions (2015)

Minions (2015) - Community 9		
KNN	Movie Name	Community
1	Dragon Age: Origins (2009)	9
2	Transformers: Dark of the Moon (2011)	17
3	Spider-Man 2 (2004)	9
4	Celebrity (1998)	9
5	Transformers: Revenge of the Fallen (2009)	9

Question 7

We now downloaded the `movie_ratings list` and use it along with the movie network to predict the ratings of the above 3 movies.

Prediction of Ratings

We processed the `movie_ratings list` as in `Question 1` and then mapped all the movie (> 5 actors/actress) indexes to subsequent ratings. Ratings predictions was done in 3 different ways:

1. `Community based` - Average ratings of all the nodes in the community, to which the node to be predicted belong as a feature.
2. `K nearest neighbors` - Average ratings of all the k nearest neighbors in the community, to which the node to be predicted belong as a feature.
3. `Based on all the neighbors` - Average ratings of all the neighbors, to which the node to be predicted is a neighbor to, as a feature.
4. `Average of all methods` - Average of all the above techniques.

Results of the prediction are as follows:

Rating	**Batman vs Superman: Dawn of Justice (2016)**	**Mission: Impossible - Rogue Nation (2015)**	**Minions (2015)**
`Actual IMDb`	7.1	7.5	6.4
`Community Based`	6.074895	6.129327	6.195559
`kNN (k=20)`	6.5	6.985714	7.226667
`All Neighbors`	6.110256	5.99537	6.396138
`All Combined`	6.07562	6.121225	6.216459

Question 8

Addition of Features

The following two features (computed in Python) were considered to construct a regression model

1. `Top 5 page ranks of the actors in each movie`: Top 5 page ranks of actors belonging to a particular movie were assigned as a vector to each movie node.
2. `101 boolean values of 100 top directors`: We downloaded director_movies.txt file, processed it to extract directors of all the movies. We created a dictionary in python of `Director` $\rightarrow$ `Movie` and then denoted each director as 1 if he is among the top 100 or else 0. We then mapped this directors to their respective movies.

Model Summary and Analysis

Once the above additional features were created we trained our linear regression model to predict ratings for the movies. The summary of the regression model is provided below,

	Dependent variable:
	Ratings
PageRank1	9,333.289
	(826.189)
PageRank2	448.005
	(2,140.270)
PageRank3	−5,274.017
	(3,558.297)
PageRank4	−17,421.140
	(4,877.013)
PageRank5	146,093.000
	(3,924.731)
Director	1.507
	(0.103)
Constant	2.539
	(0.011)
Observations	246,378
R^2	0.096
Adjusted R^2	0.096
Residual Std. Error	2.973 (df = 246371)
F Statistic	4,346.148 (df = 6; 246371)

Table 3: Regression Model Summary

Results using Linear Regression

Rating	Batman vs Superman: Dawn of Justice (2016)	Mission: Impossible Rogue Nation (2015)	Minions (2015)
Actual IMDb	7.1	7.5	6.4
Predicted	6.040144	5.870426	6.044278

Question 9

In this question we create a `bipartite graph` with one subset being actors and the other subset being movies they enacted in. Edges are only drawn between one set to another and not within each sets. The edgelist was created using Python and since the list is ordered instead of using `make.bipartite.graph` function a simple `graph.data.frame` function was used to create the bipartite graph.

Bipartite Graph Details

$$\texttt{Number of edges within the graph} = 3429608$$
$$\texttt{Number of vertices} = 487734$$

Rating of Actors

Since we already calculated the ratings in `Question 7` we simply loop through each actor and obtain its neighbors and for each neighbor (movie) the corresponding rating is located and the mean of the ratings is assigned to the actor.

Predicting Ratings of Movies

We used 3 techniques after assigning ratings to the actors to predict the ratings of the movies.

1. `Linear Model`: The `top 5 actor ratings` for each movie was found and used as the training data and the class prediction was set to the actual prediction of the movie. The predicting data comprised of the `top 5 actor ratings` of the 3 movies.
2. `Average Top 5 Ratings`: The average of the `top 5 actor ratings` for each of the 3 movies was taken to obtain the predicted ratings.
3. `Average of All Actors Ratings`: The average of all actors ratings for each of the 3 movies was taken to obtain the predicted ratings.

Prediction Results

Results of the prediction are as follows:

`Rating`	`Batman vs Superman: Dawn of Justice (2016)`	`Mission: Impossible - Rogue Nation (2015)`	`Minions (2015)`
`Actual IMDb`	7.1	7.5	6.4
`Linear Model`	6.168	6.104	6.834
`Top 5 Actors`	6.276	6.144	7.056
`All Actors`	3.929	3.763	6.022

Analysis of Results

- The `All Actors` approach gives a low value for 2 of the movies and not for `Minions (2015)` because the former two have lots of supporting actors who worked on the film, who might have lower rating score, while the latter is an animated movie and usually there aren't that many supporting actors in animated movies as most of the work is done through voice recordings. Hence `Minions (2015)` has a higher rating.
- The `Linear Model` and `Top 5 Actors` provide almost the same results for all the 3 movies.