Pic 16 Final Project Report

Minghan Li, Tingyi Lu, Xinran Qian

Introduction:

This project proposes some shallow view on the making of a successful movie by creating and analyzing an actor/actress network from IMDb data https://www.imdb.com/interfaces/, and an importance ranking of the actors and actresses. The project was written in Python 2.7 and paid particular attention to PageRank and the visualization of the network. This report explains our research problem in great detail, as well as how the program works. A pseudo-code is included so that a Python programmer could read and understand how to create the program from scratch.

Datasets Description:

For this project, we utilized the following four datasets downloaded on March 4th, 2019: name.basics.tsv, title.ratings.tsv, title.basics.tsv, and title.principals.tsv. These datasets are updated daily by IMDb, and can be downloaded from: https://datasets.imdbws.com/. Table 1. shows the variables that we used from each dataset.

Name.basics.tsv	Title.ratings.tsv	Title.basics.tsv	Title.principals.tsv
nconst primaryName	tconst averageRating numVotes	tconst startYear titleType	nconst category

Table 1. Variables used in each dataset

Name.basics.tsv

- Variables used: nconst, primaryName

This dataset contains basic information of 9,177,198 casts. Each of the personnel is uniquely identified by an identifier named "nconst". We displayed the primaryName of each cast on the final network graph. We ignored columns birthYear, deathYear, primaryProfession, and knownForTitles.

Title.ratings.tsv

- Variables used: tconst, averageRating, numVotes

This dataset contains the number of votes and average ratings of 914,862 films, which are uniquely identified by an identifier from column "tconst". We used the identifiers to locate each film in different datasets. We filtered movies by restricting averageRating of movies to be higher than or equal to 7.8/10, and numVotes more than 200.

Title.basics.tsv

- Variables used: tconst, startYear, titleType

This dataset contains basic information of 5,702,733 titles that have unique identifiers listed in "tconst". We extract titles that are classified as movies, that are produced after 2000, since our primary focus is on recent movies in which the actors/actresses are still alive. So that we can obtain a practical optimal combinations of actors/actresses to produce a good film.

Title.principals.tsv

- Variables used: nconst, category

This dataset provides more information on 32,684,118 casts. We filtered all casts and found out casts who are listed as actors/actresses.

Processed Data:

After applying multiple masks on the datasets, we obtained an actor list of size 10,316, which contains all the actors who appeared in recent, top-rated movies. As a result, we generated an adjacency matrix of dimension 10,316*10,316.

Design:

Part 1: Import Data/Modules

It is important to import all useful modules including Pandas, Numpy, Matplotlib.pyplot, and NetworkX. After importing the modules, import the raw datasets (tsv.file) listed above.

Part 2: Data Cleaning

After importing the data and modules, it is critical to tidy and clean up those datasets since they are too big and contain too much redundant information that is not useful for our research purpose. Therefore, we apply several filters to create a subset of the original datasets for us to process.

First of all, we only would like to consider movies. Then we limit their releasing years to be no earlier than 2000. Since we also would like to include their ratings into our research, we find those movies with ratings higher than 7.8 and more than 100 votes. Now, we have a list of movies.

The second step is to find each movie's corresponding list of actors and actresses.

Lastly, we create adjacency matrices based on the number of total actors/actresses, and update each element based on the number of connections (whether or not the two actors/actresses appeared in a same movie) and the ratings of the movies.

Part 3: Ratings Algorithm

To obtain meaningful rating for each actors/actresses pair, we need to take into account all the movies in which the two actors/actresses collaborated. We decided that the average ratings for all movies featuring each pair would provide a reasonable indication of overall result of having the two in the same movie. To accomplish this, we created two adjacency matrices.

Both the row and column contained all names for actors/actresses. Each element in the matrix represents a piece of information about an actors/actresses pair. The first matrix stores the sum of all ratings of movies in which the two actors/actresses appear, while the second matrix stores the number of times the two actors/actresses appear in the same movie. After filling these two matrices with values, we performed element-wise division, which yields a new matrix with average ratings between each actors/actresses pair.

Part 4: PageRank

The importance of an particular actor/actress, however, depends on multiple factors, such as the number of connections between the actor/actress and other actors/actresses, the importance of those actors/actresses to which he or she connects, and the average ratings for all the connections. We therefore decided to use PageRank algorithm of the NetworkX module to obtain the ranking of importance of all the actors/actresses. We created a NetworkX plot by importing the all actors/actresses ID numbers as nodes, and adding edges between the nodes with weights based on the average movie rating of the each actors/actresses pair. We applied PageRank algorithm to the network, which gave us the scores for all the actors/actresses. Using sort function, we obtained the ranking for all the actors/actresses included in our data set. The 20th largest score and 70th largest score would serve as two thresholds. We keep the actors with top 20 scores in a dataset and the rest 50 actors/actresses with scores between 20th largest score and 70th largest score will be kept in another dataset.

Part 5: Visualization

So far we processed our dataset only with respect to the specific 'nconst' of each actor/actress, therefore, we needed to relabel all nodes through a mapping from the 'nonst' attribute of an actor/actress to his or her actual name. The above process can be achieved using the relabel_nodes function of the NetworkX module. Then, based on our PageRank value, we removed the nodes (actors/actresses) who didn't rank top 70 from our graph. We also labeled the top 20 actors/actresses and the rest of the top 70 in different colors. Finally, we arranged the nodes in a shell layout, with the top 20 forming the inner circle and the rest 50 forming the outer circle to make our network plot more visually pleasing.

<u>Tools Description:</u>

- Pandas

  We used Pandas module to import, clean, and manipulate the datasets. In particular, we utilized function pd.merge to obtain panda dataframes containing essential information from two separate dataframes.

- PageRank

We created a weighted matrix with each entry based on the number of connections between two actors/actresses and the rating of the movie that they collaborated on. Then we ranked the importance of actors/actresses with pagerank function from module networkx with damping factor alpha set to 0.85.

- Shell layout
  We used shell layout to create an inner shell and an outer shell for the final top 70 most important actors/actresses. The inner shell highlights the top 20 actors/actresses who are marked in light blue. And the outer shell contains the rest 50 actors/actresses who are less important and are marked in grey. The connections among actors/actresses are in orange. In this way, the finalized graph is visually pleasing. One can easily identify the most important actors/actresses and their associations to formulate a cast to produce a promising movie.

Pseudo Code:

Part 1: Import Data/Modules

IMPORT useful modules: pandas, numpy, matplotlib, networkx
READ data from tsv files with "\t" as separators

Part 2: Data Cleaning

Create mask for movie released after 2000
Get filtered movie dataframe
Create mask for ratings above 7.8 with more than 100 votes
Get filtered movie ratings dataframe
MERGE movie dataframe and movie ratings dataframe to get all movies with ratings
Create mask for actors & actresses
Get filtered actors id dataframe
MERGE actors dataframe with existing data frame to get a new dataframe with all actors and actresses ids for movies with rating
Get the list of all actors ids appeared in our dataframe and convert it into a list with no replacement
MERGE name list with the new dataframe
Create a dictionary that maps actor ids to actor names

Part 3: Ratings Algorithm

Create two matrix-like data frames with both row titles and column titles consisted of actor ids:
    Create a ratings matrix zeros that would store the sum of all film rating for each combination of two actors

Create another connections matrix with zeros that would store the number of connections between the two actors

FOR each movie with ratings

Create a temporary list that stores all the actors ids appeared in the movie

FOR each actor id in the temporary list

FOR each actor stored behind the current actor

SET score equal to the score of movie

ADD score to corresponding sums for the two in ratings matrix

ADD one to corresponding connections in connections matrix

Create a matrix that stores the average ratings for each combination of two actors using connection matrix and ratings matrix

Convert the dataframe matrix into a numpy array

Part 4: PageRank

Create a networkx Digraph with notes from our filtered actor ids

FOR each actor id representing the row of our numpy matrix

FOR each column in the current row

IF the average rating between to two actors ids is greater than zero

ADD an edge between the two actor ids with weight equal to the average rating for the two actor ids

Create a shortened dictionary with only actor ids of the current nodes

Relabel the nodes with the corresponding actor names in the shortened dictionary

Apply PageRank algorithm to the network

Get the values for the result and rank them from smallest to largest

Get the 70th and 20th largest PageRank values to serve as dividing lines

Remove nodes with unrecognizable names

Remove all nodes except for the nodes with the best 70 PageRank values

Part 5: Visualization

Create an inner layout list containing the nodes with top 20 PageRank values

Create an outer layout list containing the 50 nodes with the PageRank values between the 20th largest value and 70th largest value

FOR each actor in the 70 actors left in the nodes

IF the actor belongs the the top 20

Assign color #A1D6E2 to the node

Add the node to the inner layout

IF the actor belongs the the other 50

Assign color #BCBABE to the node

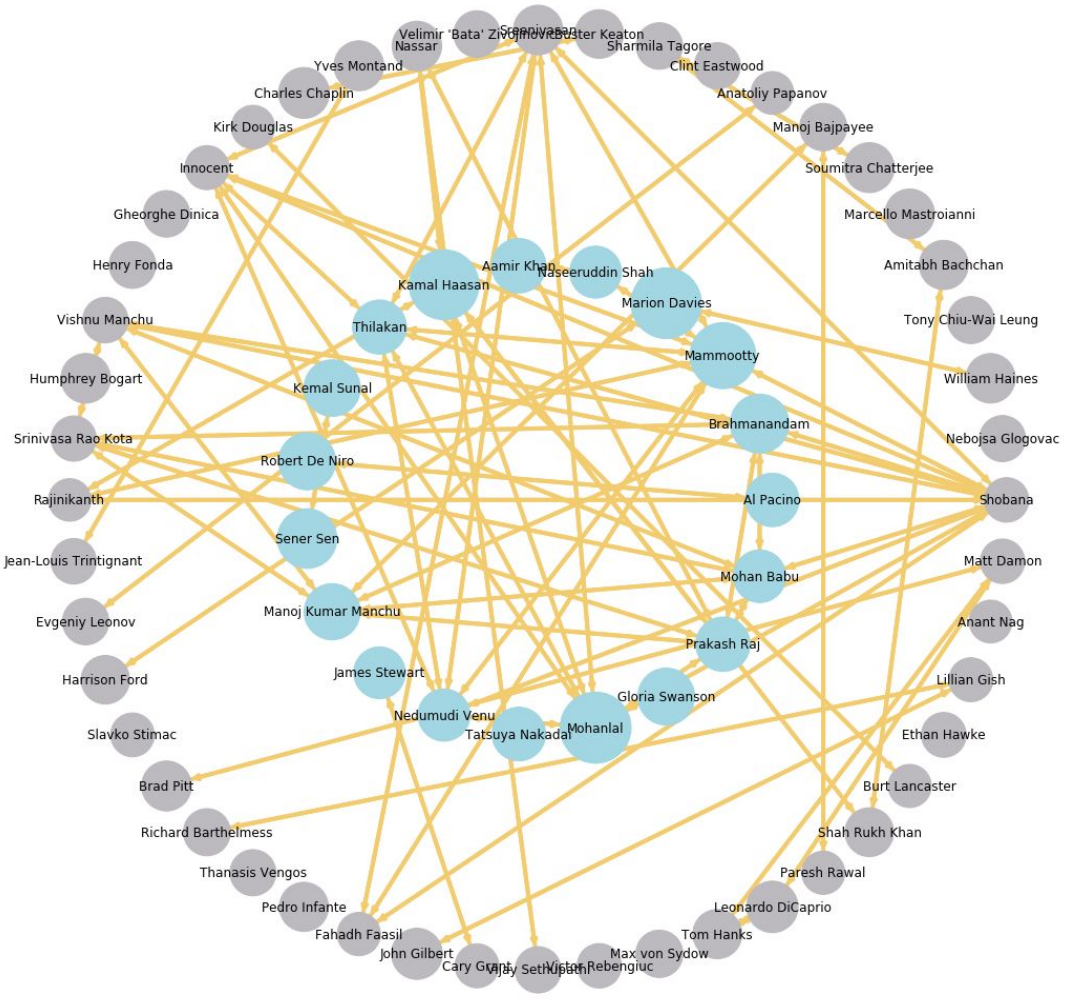
Add the node to the outer layout

SET the plot size to be 15x15
SET the graph layout as shell layout with inner circle consisting of top 20 actors and outer circle consisting of the other 50 actors
Draw the network Digraph with the following settings:
labels of font size 12, edges with width of 4 and color #f4cc70
Show plot

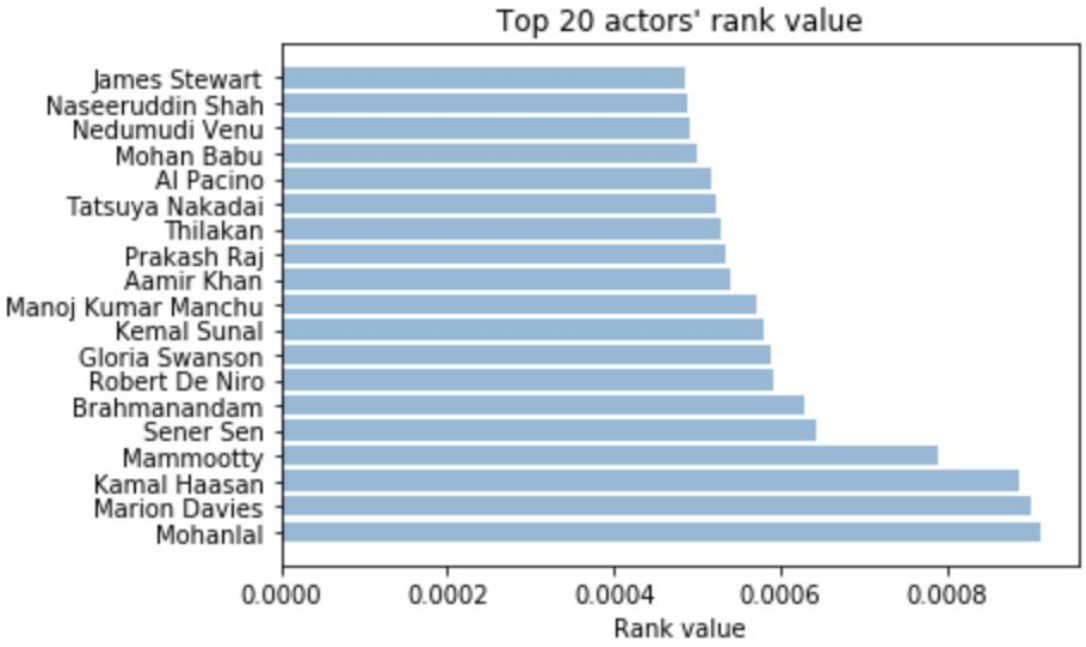
Discussion:

Visualization

Because the entire network contained over 10000 actors/actresses, it was impossible to visualize all of them on a single plot. In fact, it would also be quite meaningless to include those actors/actresses who have low ranks. Therefore, we decided to include only the most important 70 actors/actresses in our visualization of their network. We used NetworkX and matplotlib.pyplot in our visualization.



The above graph is our network. There are a total of 70 nodes each representing an actor/actress: the top 20 forms the inner circle, while the rest 50 forms the outer circle. Each node has a different size that is proportional to its PageRank value. Each edge between two nodes represents the co-appearance of both actors/actresses in one movie. Some nodes may have a lot of edges, while some may not have any. The absence of edges for some nodes suggests that even though these actors/actresses don't have any mentionable collaborations with other top 70 actors/actresses, they still managed to make their movies successful.



To demonstrate the numerical result of our research, we made a horizontal bar plot as above. From the graph, one can clearly identify the most important actor by our criteria is Mohanlal, along with Mammootty (ranked 4th in our research result) are the biggest stars in Indian Malayalam movies. The 20th most important actor is James Stewart.

Interpretation

Since our research topic is the making of a successful movie, the abundance of edges within our network suggests that the collaboration between important actors/actresses can contribute a lot to the quality and high rating of their movies. However, there are also exceptions as some actors/actresses on their own can make their movies successful. For example, Aamir Khan is among the top 20 of our 10000+-entry actor/actress list, but he does not have any collaboration with any of the other top 70 most important actors/actresses. Therefore, our network suggests that the collaborations between top actors/actresses do contribute to, but may not be necessary for, the making of a successful movie; sometimes a single actor/actress alone can make the movie successful.

Another interesting fact we've noticed from our visualization is the surprisingly high percentage of Indian actors/actresses. This may suggest that Indian people are very fond of their movie culture and therefore tend to give high ratings to their own movies.

In the scope of our data and network, we boldly propose that over the past twenty years, movies with the above 70 actors/actresses, with or without them collaborating with each other, tend to be quite successful (with IMDb rating >= 7.8). However, since the majority of the top listed actors/actresses are gradually fading away from the movie industry and the young blood is flowing in, our proposal on the making of a successful movie may not be applicable in the future.

Limitation

First of all, after the initial filtering, the actors list we obtained contained three names with special characters. Those names can not be properly displayed in the final graph. We ended up removing those names from the actors list, even though they may have produced movies with perfect ratings.

Moreover, IMDb constantly updates its data on the website. As a result, the files downloaded on different dates may contain different data. We resolved this issue by constantly updating our datasets. However, IMDb may sometimes only update a particular file, causing a mismatch between two data files. For instance, when we tried to merge the name list with our filtered movies list, some actors were eliminated from the list because they somehow did not have identification codes. This may have caused some potential candidates to be ignored.

Furthermore, as IMDb is our only data source, there is a potential that our data is incomprehensive. There are other reputable online databases, such as Rotten Tomatoes, that we can also incorporate to produce more comprehensive and objective datasets. Also, there might large set of movie viewers who do not use IMDb to rate films, such as viewers from China. There could be great Chinese movies that are under or not rated on IMDb that we overlooked.

Another problem we encountered during our research process was the physical limitations of our hardware. Our computers failed to execute many chunks of our code when the dimension of our cleaned dataset still exceeded its limitation. Therefore, we tried quite a few different filters on the original dataset in order to narrow down the amount of data we would eventually analyze on, as you can see from the weird 7.8 cut-off point for movie ratings.

Improvement

Due to time constraints and hardware limitations, we had to impose quite a few filters on the IMDb data in order to process them on our laptops. Further research can be done on a larger data set so that the results may be more reliable. What's more, since we broadly included movies all over the world in our research, geographical factors may also be considered in the future. Furthermore, we could take in consideration more ratings from other online databases such as Rotten Tomatoes to obtain more comprehensive data source.

Conclusion:

In this project, we imported and cleaned huge datasets using Pandas and numpy modules. We applied masks to only include movies which were produced later than 2000 and rated higher than or equal to 7.8 on IMDb. And we visualized the result of our research by using networkX to create a shell layout graph that indicates the top 70 most important actors/actresses. From the above visualizations and analysis, we believe that movies with the top 70 actors/actresses tend to obtain higher ratings than the others. The numerous edges further indicates that collaborations among these actors/actresses tend to produce highly-rated films. Our research can be further improved by overcoming the limitations in computer hardware, initiating more detailed analysis based on different regions and languages and incorporate more source of data.