**1. Merge and filter the data**

*Preprocessing data(optional)*

In this exercise we used the following text files to create network.

● Actor\_movies.txt

● Actresses\_movies.txt

These file have entries for all the movies each actors and actresses have acted, voice or credited.

But these file has some Latin letters which may cause errors in some compile environments, so if there are some errors about the form, we can implement our ‘preprocessing.R’ to delete the Latin letters. But it may cause the loss of information so it is optional.

*Merge the 2 datasets and filter the people with less than 10 movies*

The objective of the exercise was to create a single file with movies entries of

each actor and actress who has acted in more than 4 movies. But since the amount of data is too large which will cause the disk crash, we set the threshold to more than 9 movies.

Thus we first use Python to merge both the text files and then removed the actors and actresses with less than 10 movies.

**2. Construct the weighted and directed graph**

In this problem we constructed a weighted graph G(V,E) using the data set created in the first problem. The parameters of the graph are defined as follows:

● Vertex (V) has all the actors and actresses name

● Si = {m|i ∈ V, m is a movie in which i has acted}, i.e. set of movies an actor/actress has acted.

● Edge (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.

Similar to the homework 3 problem set, the data frame was structured as three columns, the first column had the vertex from where the edge is starting and the second column is the terminal of the edge and the third column had the weight of the edge.

Moreover, we also used the following algorithm to efficiently process the large data frame and create the network:

1. Create a [MOVIE TO ACTOR] dictionary with 'movies' as the keys and 'set of actors who have acted in that movie' as the values.

2. For each movie in [MOVIE TO ACTOR]: Find combinations of all actors. This will give us n^2 combinations where n is the number of actors who have acted in that particular movie.

3. Then we create a [ACTOR ACTOR] dictionary with '(actor actor) combination' as the keys and the 'number of movies those two actors have acted together in' as the values.

4. For each movie in [MOVIETOACTOR]: For each (actor actor) combination in the movie list: Initialize the key of [ACTOR ACTOR] dictionary to 1 if first encounter.

Add 1 to the key of [ACTOR ACTOR] dictionary if movie already encountered.

**3. Page rank for Top 10**

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.

The pagerank algorithm with damping factor = 0.85 was run on the weighted actors’ graph and the names of the top 10 actors were determined.

|  |  |  |
| --- | --- | --- |
| **No.** | **Actor/Actress** | **Page Rank** |
| 1. | Flowers, Bess | 2.3576e-04 |
| 2. | Tatasciore, Fred | 2.038723e-04 |
| 3. | Harris, Sam (II) | 1.977247e-04 |
| 4. | Blum, Steve (IX) | 1.975e-04 |
| 5. | Miller, Harold (I) | 1.730496e-04 |
| 6. | Jeremy, Ron | 1.592508e-04 |
| 7. | Lowenthal, Yuri | 1.588602e-04 |
| 8. | Phelps, Lee (I) | 1.575774e-04 |
| 9. | Downes, Robin Atkin | 1.532819e-04 |
| 10. | O'Connor, Frank (I) | 1.471701e-04 |

When searching these top 10 actors/actresses on the Internet we found that most of them have appeared in many movies. These actors/actresses have also appeared in movies as supporting roles. Since PageRank algorithm score depends on the degree of the nodes, these actors/actresses have very high score due to large number of movies they acted, as well as shared a lot of movies with other actors/actresses. We can also observe that most of the actors/actresses are from early 1900, where actors/actresses used to act in lot of movies unlike today.

Below is top 10 famous actors/actresses in our opinions.

|  |  |  |
| --- | --- | --- |
| **No.** | **Actor/Actress** | **Page Rank** |
| 1. | Tom Cruise | 1.172932e-05 |
| 2. | Natalie Portman | 4.400885e-06 |
| 3. | George Clooney | 2.195821e-06 |
| 4. | Kate Winslet | 1.020989e-05 |
| 5. | Robert Downey Jr. | 7.395434e-06 |
| 6. | Leonardo DiCaprio | 1.960902e-06 |
| 7. | Brad Pitt | 1.474613e-06 |
| 8. | Julia Roberts | 2.101313e-06 |
| 9. | Johnny Depp | 4.085536e-06 |
| 10. | Sandra Bullock | 1.200901e-06 |

Apparently the famous actors/actresses in our opinion has very low PageRank score compared to the top 10 actors/actresses based on PageRank score. These

actors/actresses have appeared in much less movies as lead role as well as in supporting roles compared to the top 10 who have many appearance, as a result they have high node degree in the concerned network. As a result, this anomaly is justifiable in a sense and we can observe that PageRank is not an effective measure for calculating actors/actresses popularity.

*Significant Pairing*

We analyzed the network for any significant pairing of actors/actresses who have acted together most of the time. For these pairing nodes they have directed edges between them in both directions and have weight of 1 so that we all the movies acted by any one of them must also be acted by the other. We found more than 51 such pairings of nodes (actors/actresses) and list some of them as following:

|  |  |
| --- | --- |
| Actor/actress parings | Description |
| Randi Brough and Candi Brough | Twin sisters and have acted only in movies together. |
| Marlon Wessel and Leon Wessel Masannek | They always act the same movies. |
| Matthew Salinas and William Brush | They just act the same comedy series. |

What we can learn from the above table is that pairings happen because they are siblings or they only act a very few movies or series of programs or they belong to the same company which is very small.

**4. Construct the movie graph**

In this part we just used the data text files referred in problem 1 to create the movie network. We constructed a weighted graph G(V,E) using the data set created in the problem 1. The parameters of the graph are defined as follows:

● Vertex (V) has all the movies name

● We only considered movies which has at least 10 actors/actresses

● Edge (E) = {(Mi, Mj)}, i.e. an edge is created if two movies (vertices) Mi and Mj which have A set of actors/actresses and B set of actors/actresses respectively.

● Weight (W) of each edge (i to j) was assigned as W = |A ∩ B|/ |A U B|, i.e. ratio of the set of actors/actress acted in movies Mi and Mj to the set of total actors/actresses of movies Mi and Mj. This ratio is known as jaccard index, which makes the graph simple and undirected.

*Creating the movie network*

As the network is undirected, we just created a data frame for creating the network. Similar to the homework 3 problem set, the data frame was structured as three columns, the first column had the vertex from where the edge is starting and the second column is the terminal of the edge and the third column is the weight of the edge.

Then we implemented read.graph of igraph to construct the graph.