The Argentine Movie Actors Network

Social Network Analysis (University of Michigan, Autumn 2013)

COURSERA

Teacher: Lada Adamic

Student: Walter Marcelo Lamagna (wlamagna@gmail.com)

Option 1: empirical network analysis.

Question: Task: find data, analyze data (and visualize it), then interpret.
Please upload a pdf of your answers in the text box below. Below the answer box, you can see the grading rubric that will be used during the evaluation phase to help you plan your answer accordingly. You're not expected to fill it out during the submission phase.

Index

1) Introduction

Argentina has an active movie actors community. In this work a connection between actors exists if both actors have been together in a movie / or tv show chapter. The dataset was created from scratch.

First the list of Argentine movies and tv shows from 2001 to 2013 was obtained from Wikipedia (table #1).

Secondly the list of actors was obtained from imdb, the actors that where in less than 6 movies and or tv. shows, have been removed.

For the remaining list of actors (table #4), all the movies and their list of actors where obtained, and finally, a network was buid where an edge between two actors mean that they where together in a movie/tv show.

The edges are weighted based on the amount of movies or tv shows when they where together (independently if they interacted during the movie).

The network has 316 nodes (Actors) and 2164 edges.

2) Obtaining data

The data acquisition has many technical details because the network was created from scratch. The scripts are in the section “Source Code”, all the data was obtained from two places:

1) Wikipedia
2) IMDB.com

From wikipedia where obtained which movies where done in Argentina in the period 2001 – 2013. The movies where obtained in this URL:
http://es.wikipedia.org/wiki/Anexo:Pel%C3%ADculas_argentinas_de_2013

The Wikipedia webpage has this information:

Anexo:Películas argentinas de 2013

Esta es una **lista de películas argentinas estrenadas durante 2013**

Películas argentinas de 2013			
Título	**Director**	**Estreno**	**Género**
A - C			
Aire de chacarera	Nicolás Tacconi	15 de agosto	
Antes	Daniel Gimelberg	7 de marzo	
El árbol de la muralla	Tomás Lipgot	14 de febrero	
Beirut - Buenos Aires - Beirut	Hernán Belón	28 de marzo	
Buscadores de identidades robadas	Miguel Rodríguez Arias	19 de septiembre	
Caídos del mapa	Nicolás Silbert y Leandro Mark	19 de septiembre	
Calles de la memoria	Carmen Guarini	4 de julio	

(table #1)

The title was copied into a text file and with a script, its content looks like this:

```
walter@walter-Lenovo-Z480:~/Desktop/Peliculas$ cat titulos.txt
Aire de chacarera
Antes
El árbol de la muralla
Beirut - Buenos Aires - Beirut
Buscadores de identidades robadas
Caídos del mapa
walter@walter-Lenovo-Z480:~/Desktop/Peliculas$
```

(picture #1)

The movies title search in IMDB:

```
walter@walter-Lenovo-Z480:~/Desktop/Peliculas$ ./buscar_imdb.sh
tt3127056       Aire de chacarera
tt1636422       Antes
tt2641162       El árbol de la muralla
tt2175573       Beirut - Buenos Aires - Beirut
        Buscadores de identidades robadas
tt3173396       Caídos del mapa
```

(picture #2)

At this point, there was acquired this data:

YEAR	MOVIE TITLE WIKIPEDIA	DIRECTOR	IMDB id	MOVIE Title From IMDB
2013	Aire de chacarera	Nicolás Tacconi	tt3127056	Aire de chacarera
2013	Antes	Daniel Gimelberg	tt1636422	Antes
2013	El árbol de la muralla	Tomás Lipgot	tt2641162	El árbol de la muralla
2013	Beirut - Buenos Aires - Beirut	Hernán Belón	tt2175573	Beirut - Buenos Aires - Beirut
2013	Buscadores de identidades robadas	Miguel Rodríguez Arias		Buscadores de identidades robadas
2013	Caídos del mapa	Nicolás Silbert y Leandro Mark	tt3173396	Caídos del mapa

(table #2)

Some titles were not found, if they couldn't be found manually, the movie title was excluded from the dataset. Now, using the “IMDB id” the Director's Name was searched, this helped to gain confidence that the movie title obtained from IMDB was the same than the found in Wikipedia. To do this the “IMDB id” is writtein in a file and then another program brings the data from IMDB:

```
walter@walter-Lenovo-Z480:~/Desktop/Peliculas$ head res.txt
tt3127056
tt1636422
tt2641162
tt2175573
tt3173396
walter@walter-Lenovo-Z480:~/Desktop/Peliculas$
```

(picture #3)

```
walter@walter-Lenovo-Z480:~/Desktop/Peliculas$ ./buscar_detalles.sh
tt3127056       ;Documentary    Argentina;      2013-08-15      Aire de chacarera  (2012)       ;Nicolás Tacconi        ;Diego Arnedo (nm5654
431);Fernando Arnedo (nm5883289);Fernando Arnedo (nm5883288);Ronnie Beltrán (nm5883291);Alberto Bravo de Zamor (nm5883290);Elpidio Herrera (n
m1370926);Juan Saavedra (nm1437354);Juan Cruz Suárez (nm5883292);Morenito Suárez (nm5883293);Víctor Manuel Ábalos (nm5883294)
tt1636422               Argentina;      2013-03-07      Antes  (2010)   ;Daniel Gimelberg       ;Gabino Acosta (nm3875612);Horacio Acosta (nm
1297835);Jorge Booth (nm1878120);Guadalupe Docampo (nm1941751);Alejandra Flechner (nm0281371);Verónica Llinás (nm0515786);Carlos Portaluppi (
nm0691937);Nahuel Pérez Biscayart (nm1465580);Nahuel Viale (nm2062733)
tt2641162       ;Documentary    Argentina;      2013-02-14      El árbol de la muralla  (2013)  ;Tomás Lipgot   ;Jack Fuchs (nm1693672)
tt2175573       ;Documentary    Argentina;      2012    Beirut Buenos Aires Beirut  (2012 TV Movie)     ;Hernán Belón   ;Grace Spinelli (nm48
47568)
tt3173396               Argentina;      2013-09-26      Caídos del mapa  (2013)        ;Leandro Mark;Nicolás Silbert   ;Eugenia Alonso (nm20
91633);Silvina Bosco (nm0097845);Ailén Caffieri (nm5931773);Sofía Calzetti (nm5931774);Tomás Carullo Lizzio (nm4975899);Brenda Marks Cobas (n
m5931775);Felipe Corrado (nm5931776);Osqui Guzmán (nm4096176);Karina K. (nm0433863);Alejandro Paker (nm4188790);Atilio Pozzobon (nm1015451);M
arcelo Savignone (nm2198365);Tina Serrano (nm0785627)
```

(picture #4)

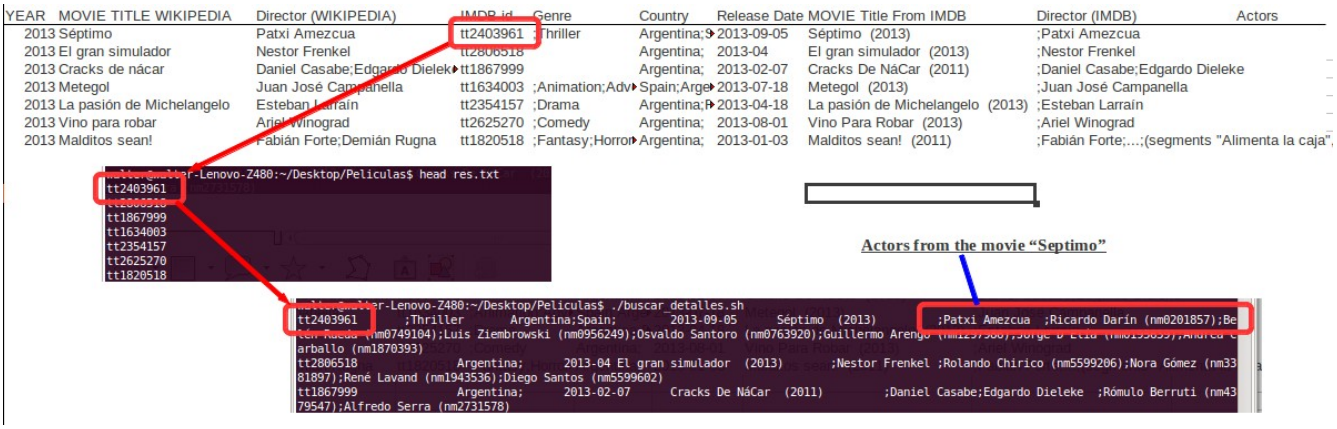
Part of the table with the movies can be observed in the table #3.

YEAR	MOVIE TITLE WIKIPEDIA	Director (WIKIPEDIA)	IMDB id	Genre	Country	Release Date MOVIE Title From IMDB	Director (IMDB)	Actors
2013	Aire de chacarera	Nicolás Tacconi	tt3127056	;Documentary	Argentina;	2013-08-15 Aire de chacarera (2012)	;Nicolás Tacconi	;Diego Arnedo (n
2013	Antes	Daniel Gimelberg	tt1636422		Argentina;	2013-03-07 Antes (2010)	;Daniel Gimelberg	;Gabino Acosta (
2013	El árbol de la muralla	Tomás Lipgot	tt2641162	;Documentary	Argentina;	2013-02-14 El árbol de la muralla (2013)	;Tomás Lipgot	;Jack Fuchs (nm
2013	Beirut - Buenos Aires - Beirut	Hernán Belón	tt2175573	;Documentary	Argentina;	2012 Beirut Buenos Aires Beirut (2012 TV Movie)	;Hernán Belón	;Grace Spinelli (n
2013	Caídos del mapa	Nicolás Silbert y Leandro Mark	tt3173396		Argentina;	2013-09-26 Caídos del mapa (2013)	;Leandro Mark;Nicolás Silbert	;Eugenia Alonso

(table #3)

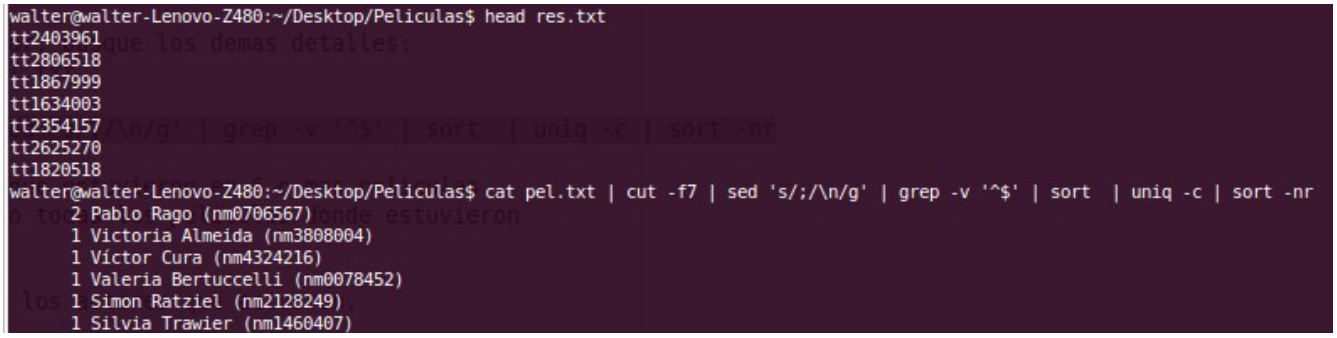
With an Open Office Spreadsheet formula, the fields “Director (Wikipedia)” and “Director (IMDB)” where compared if they match. In the examples from table #3, all they match, that means that the movie obtained from Wikipedia is exactly the movie I was looking for. If this is not the case, it was fixed manually searching for it in IMDB. Next, all the documentary genre was deleted, because they have few or none actors. Once the data was cleaned, the actor names were counted to keep the actors that participated in 6 or more movies.

The IMDB id from all the movies should be copied inside a file, and then the script to obtain the movies data is re-run. The picture #4 shows the IMDB id in the Excel file, that is written in the “res.txt” file and after running the “buscar_detalles.sh” script, it gets the movie data from IMDB.



(picture #4)

The output from the script that looks for the actors is redirected into a file, and then the actors are counted the time they appear in different movies. The actual command used can be looked in the picture #5.



(picture #5)

The actors that appear in 6 or more movies are kept, this is the resulting list:

Movies	Actor Name (imdb id)	Movies	Actor Name (imdb id)
12	Norma Aleandro (nm0001903)	7	Martina Juncadella (nm2255289)
11	Alejandro Awada (nm0043389)	7	Sergio Boris (nm0096815)
10	Ana Celentano (nm1014571)	7	Susana Varela (nm1253593)
10	Daniel Fanego (nm0266723)	6	Alberto Ajaka (nm3196782)
10	Juan Palomino (nm0658748)	6	Atilio Pozzobon (nm1015451)
10	Nicolás Condito (nm1434953)	6	Daniel Valenzuela (nm0884439)
9	Ailín Salas (nm2764483)	6	Erica Rivas (nm0729050)
9	Luis Luque (nm0527045)	6	Fabián Arenillas (nm0034330)
9	Luis Machín (nm0532721)	6	Guadalupe Docampo (nm1941751)
8	Arturo Goetz (nm0324496)	6	Inés Efron (nm2073156)
8	Guillermo Pfening (nm1166977)	6	Lautaro Delgado (nm0216992)
8	Gustavo Garzón (nm0308991)	6	Leonardo Sbaraglia (nm0768614)
8	Juan Minujín (nm1375877)	6	Martín Piroyansky (nm0685289)
8	Luis Ziembrowski (nm0956249)	6	Osmar Núñez (nm1881676)
8	Ricardo Darín (nm0201857)	6	Pablo Rago (nm0706567)
7	Carlos Kaspar (nm0440699)	6	Rafael Spregelburd (nm1041596)
7	Fernán Mirás (nm0592625)	6	Rita Cortese (nm0181300)
7	Gabriel Goity (nm0324813)	6	Victoria Carreras (nm0140262)

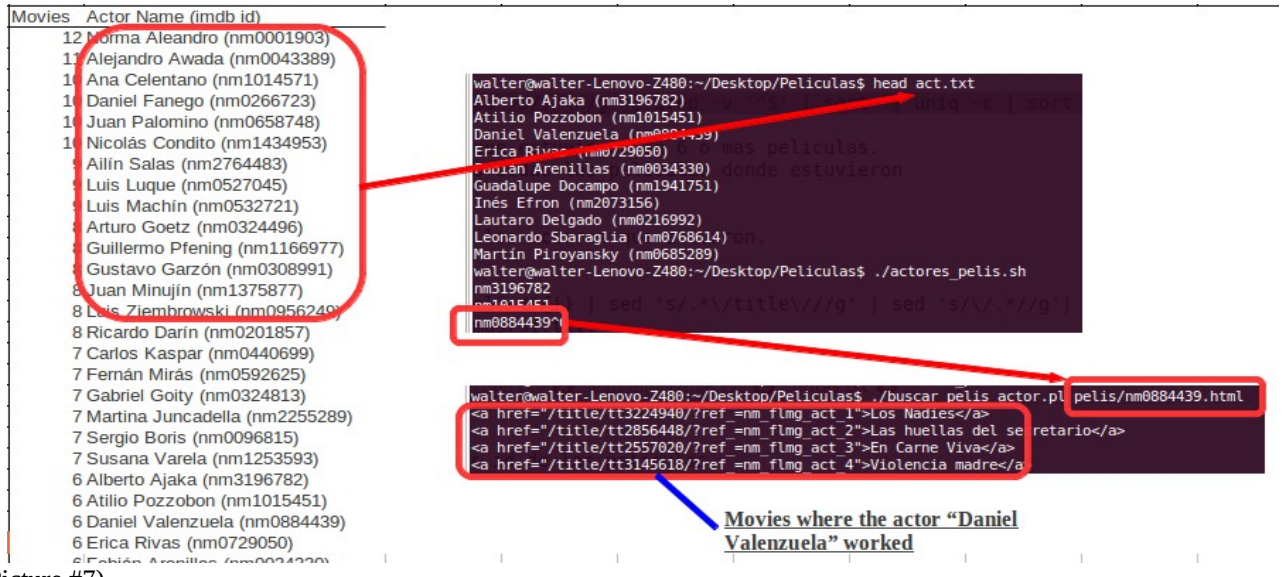
(table #4)

This IMDB id from the actors that appear almost 6 times are put into a file, and then the script "actores_pelis.sh" is re-executed (picture #6).

```
walter@walter-Lenovo-Z480:~/Desktop/Peliculas$ head act.txt
Alberto Ajaka (nm3196782)
Atilio Pozzobon (nm1015451)
Daniel Valenzuela (nm0884439)
Erica Rivas (nm0729050)
Fabián Arenillas (nm0034330)
Guadalupe Docampo (nm1941751)
Inés Efron (nm2073156)
Lautaro Delgado (nm0216992)
Leonardo Sbaraglia (nm0768614)
Martín Piroyansky (nm0685289)
walter@walter-Lenovo-Z480:~/Desktop/Peliculas$ ./actores_pelis.sh
nm3196782
nm1015451
nm0884439^C
```

(Picture #6)

The picture #7 displays the movie actors that appear in more than 6 movies/tv shows (left), their IMDB id is written in the file act.txt, and then the script "actores_pelis.sh" is executed (right), this script creates one file per actor and inside each file are all the movies where he appeared. Finally, the script "buscar_pelis_actor.pl" is executed with each of these files as argument. The source code #1 in the next page describes it further.



(Picture #7)

And now, for each movie all the actors are acquired. There will be movies where two actors worked, only one instance from each movie will be used, in this step that is done:

```
----- Start of Source Code #1 ------

> res.txt

for i in `ls pelis`; do

./buscar_pelis_actor.pl pelis/${i} | sed 's/.*\/title\///g' | sed 's/\/.*//g'| sed 's/".*//g' >>
res.txt;

done

cat res.txt | sort | uniq > /tmp/res.txt;

mv /tmp/res.txt res.txt;

./buscar_detalles.sh > demas/movies.txt;

cat demas/movies.txt | cut -f1,7 | sort | uniq | sed 's/ (nm[0-9]*)//g' | cut -f2 | sed 's/^;//g' >
per2.txt

------ End Of Source Code #1 -------
```

```
walter@walter-Lenovo-Z480:~/Desktop/Peliculas$ cat demas/movies.txt | cut -f1,7 | sort | uniq | sed 's/ (nm[0-9]*)//g' | cut -f2 | sed 's/^;/
/g' > per2.txt
walter@walter-Lenovo-Z480:~/Desktop/Peliculas$ head per2.txt
Tita Merello;Jorge Salcedo;Luis Tasca;Gloria Ferrandiz;Marta Cipriano;Juan Carlos Altavista;Oscar Villa;Adolfo García Grau;Jorge De La Riestr
a;Alberto Olmedo;Atilio Pozzobon;Alfonso Pícaro
Violeta Antier;Fernanda Mistral;Alberto Argibay;Milagros de la Vega;Norma Aleandro;Jorge Rivera López;Osvaldo Terranova;Margarita Corona;Cipe
 Lincovsky;Pedro Buchardo;Juan Manuel Tenuta
Niní Marshall;Ubaldo Martínez;Rafael Carret;Luis Tasca;Tristán;Nathán Pinzón;Ramona Galarza;Mario Lozano;Virginia Amestoy;Estela Vidal;Osvald
o Canónico;Juan Carlos Palma;Enrique San Miguel;José Ruzzo;Luis García Bosch;Alfonso Pisano;Oscar Pedemonti;Hernán Figueroa Reyes;Atilio Pozz
obon;María de los Ángeles Medrano;Alfonso Pícaro;Jorge Cafrune;Mariángeles
Palito Ortega;Sonia Bruno;Eddie Pequenino;Tono Andreu;Mariquita Gallegos;Juan Carlos Altavista;Guido Gorgatti;Rosalía;Enrique San Miguel;Carl
os Víctor Andriss;Ignacio de Soroa;Marta Cipriano;Atilio Pozzobon;José Ruzo;Karina;Tatave Moulin;Catalina Speroni;Aída Luz;Fredy Tadeo;Rafael
 Diserio;Nené Morales
```

(Picture #8)

The movies.txt file has 892 tv shows in total with its actors, genre, director, genre and year. The file "per2.txt" has all the actors for the movies, and the movies are counted only once.

```
my @aparecen = ();
foreach my $k (keys %edges) {
        my ($a,$b) = split (/,/, $k);
        my $c = $edges{"$k"};
        next if ($c < 6);
                if (! ( grep { $_ eq "$a"} @aparecen)) {
                        push (@aparecen, $a);
                }
                if (! ( grep { $_ eq "$b"} @aparecen)) {
                        push (@aparecen, $b);
                }
}

foreach my $k (keys %actores) {
        if ( (grep({$_ eq $actores{"$k"} } @aparecen))) {
                print "\tnode [\n\tid " . $actores{"$k"} . "\n\tlabel \"$k\"\n]\n";
        }
}

foreach my $k (keys %edges) {
        my ($a,$b) = split (/,/, $k);
        my $c = $edges{"$k"};
        next if ($c < 6);
        print "\tsource $a\n\ttarget $b\n\tweight $c\n\t]\n";
}
print "\n]";
close F;
~
```

(picture #9)

The Network file format choosen was gml. The network file format is created with the mat2.pl script, with this command. The script code can be found in the "Source Code" section. This command is used to create the gml file:

```
./mat2.pl > red3.gml
```

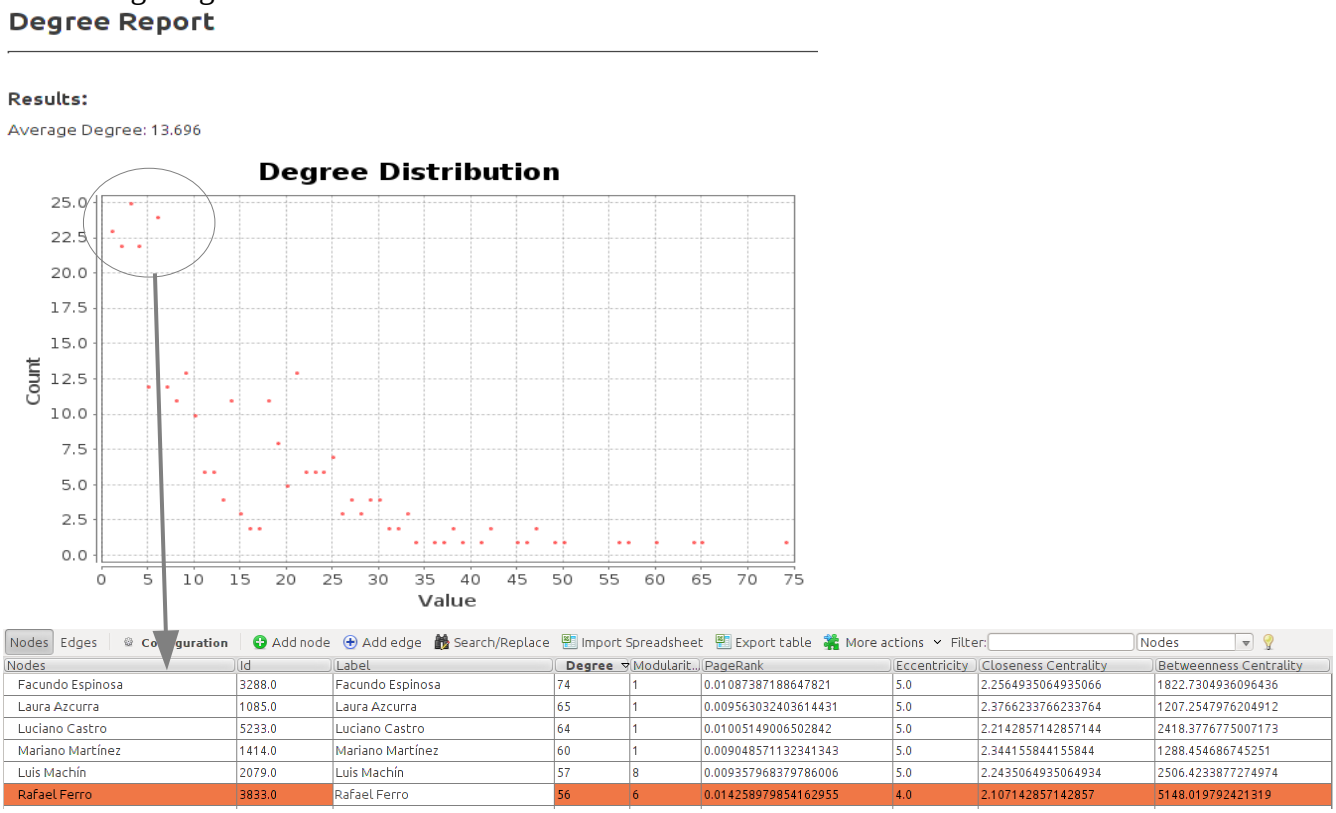
The edges weight represents the amount of times an actor has worked together with another actor in a movie. It keeps only nodes with edges with weight greater than 6. This threshold is displayed in the picture #9.

3) Data Analysis

The network is undirected and has 316 Nodes and 2164 Edges.

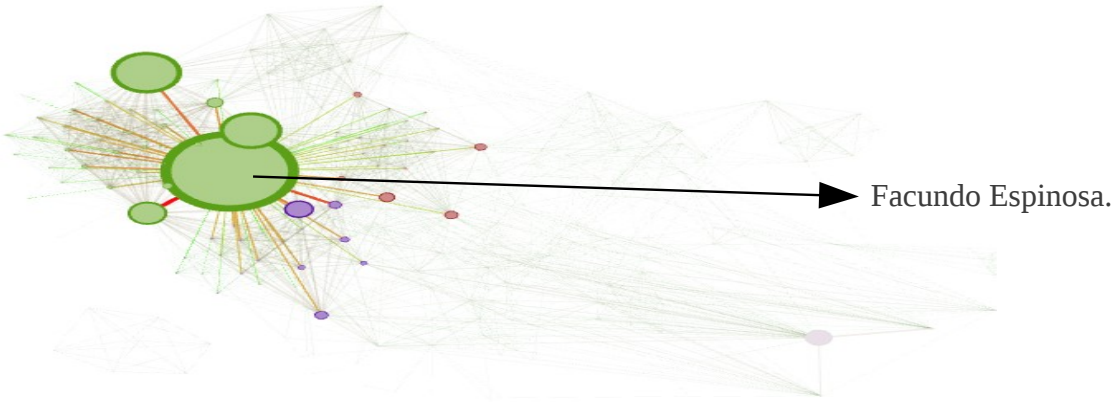
a) Average Degree

The average degree is 13.696



(Picture #10)

Using an exponential function the sizes of nodes have been set proportional to their degree. Using this method it was easy to visualy identify the highest degree nodes. A high degree in this network means that the actor participated with many other actors. This could have been because the tv show had many seasons or many actors.



(Picture #11)

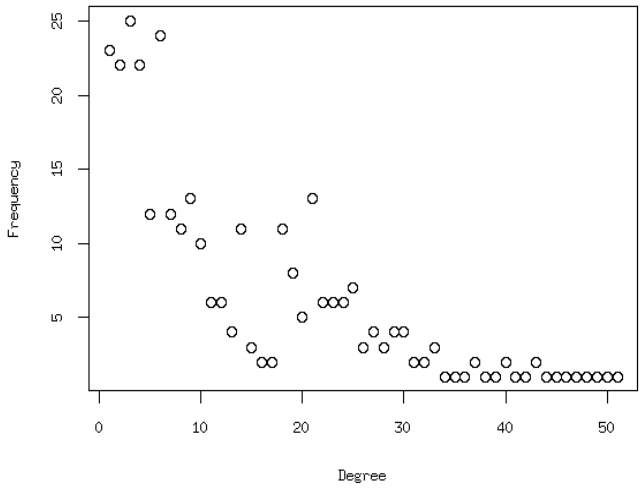
The actor **Facundo Espinosa** [10] may have in this network a high degree because he did a

variety of different characters being part of important tv shows. He was in the Tv Show "**Los Roldán (TV Series)**" [2] and "**Son amores (TV Series)**" [3], two high rating tv shows. In addition, he was in "**Campeones de la vida (TV Series)**" [1]. The other actors also have a high degree but the difference with Facundo Espinosa is considerable.

b) Power Law Distribution

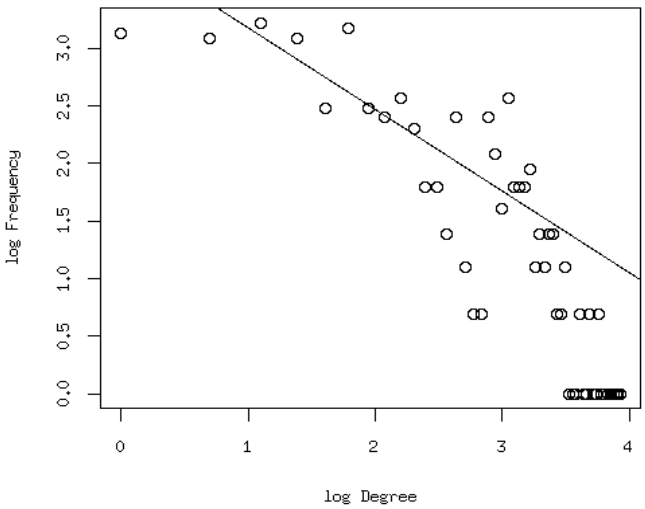
To determine if the network is power law distributed the software R was used.

```
# Reading the Network
library(igraph)
g = read.graph("red3.gml",format="gml")
d <- degree(g)
# Plot of Degree and Frequency
mm <- table(d)
degree_val <-as.numeric(data.frame(mm)$d)
degree_count <- data.frame(mm)$Freq
plot(degree_val, degree_count, xlab="Degree", ylab="Frequency")
```



(picture #12)

```
# The frequency and degree in a log-log plot.  Fitting a stright line to obtain the slope (alpha)
mm <- table(d)
degree_val <- as.numeric(data.frame(mm)$d)
degree_count <- data.frame(mm)$Freq
plot(log(degree_val), log(degree_count), xlab="log Degree", ylab="log Frequency")
abline ( lm(log(degree_val) ~ log(degree_count))
```



With a log-log plot of the degree and its frequency it <u>do not</u> follows a power law distribution.

(picture #13)

c) Network Layout

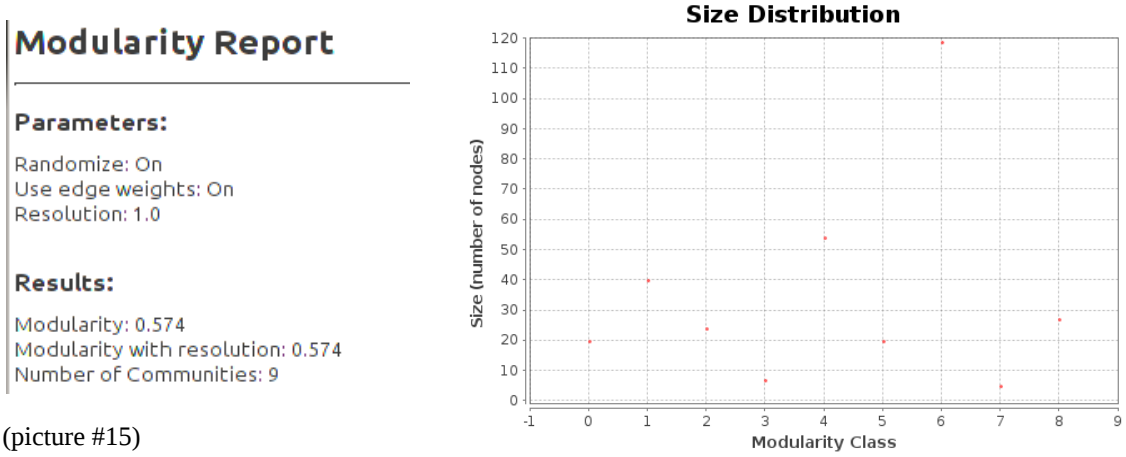
Force Atlas	
Inertia	0.1
Repulsion strength	80.0
Attraction strength	10.0
Maximum displacement	10.0
Auto stabilize function	☑
Autostab Strength	80.0
Autostab sensibility	0.2
Gravity	10.0
Attraction Distrib.	☑
Adjust by Sizes	☑
Speed	1.0

For the Layout the ForceAtlas was used, the picture 14 has the arguments used.

(picture #14)

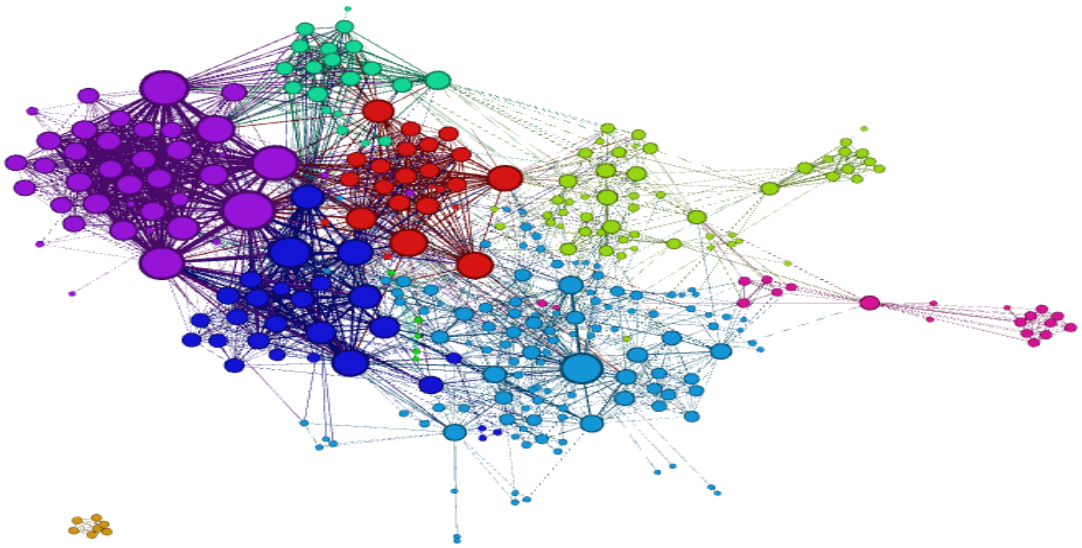
d) Finding Communities

Using gephi and the Modularity Algorithm, 9 communities were found.

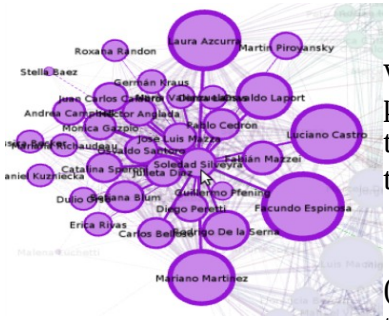


(picture #15)

A color is assigned to each community.



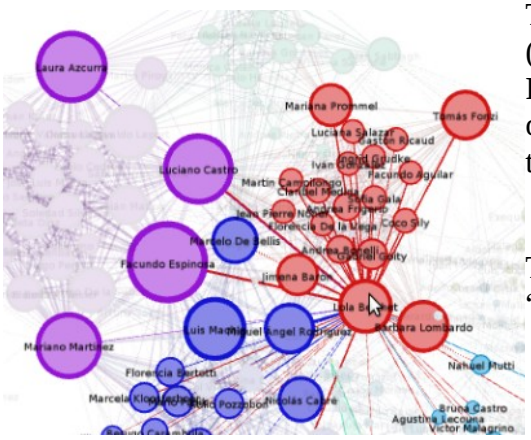
(Picture #16)



The 9 communities correspond to tv series, each has a color. This was validated by searching the actors and they match the ones that participated. The actors with larger degree and weight in their links are the ones that had a more important role in the tv show. This is because they were in more chapters.

The purple community is the tv show “Campeones de la vida (1999)” (Champions of Life) [1]. This was a tv show with two seasons that had a very large audience.

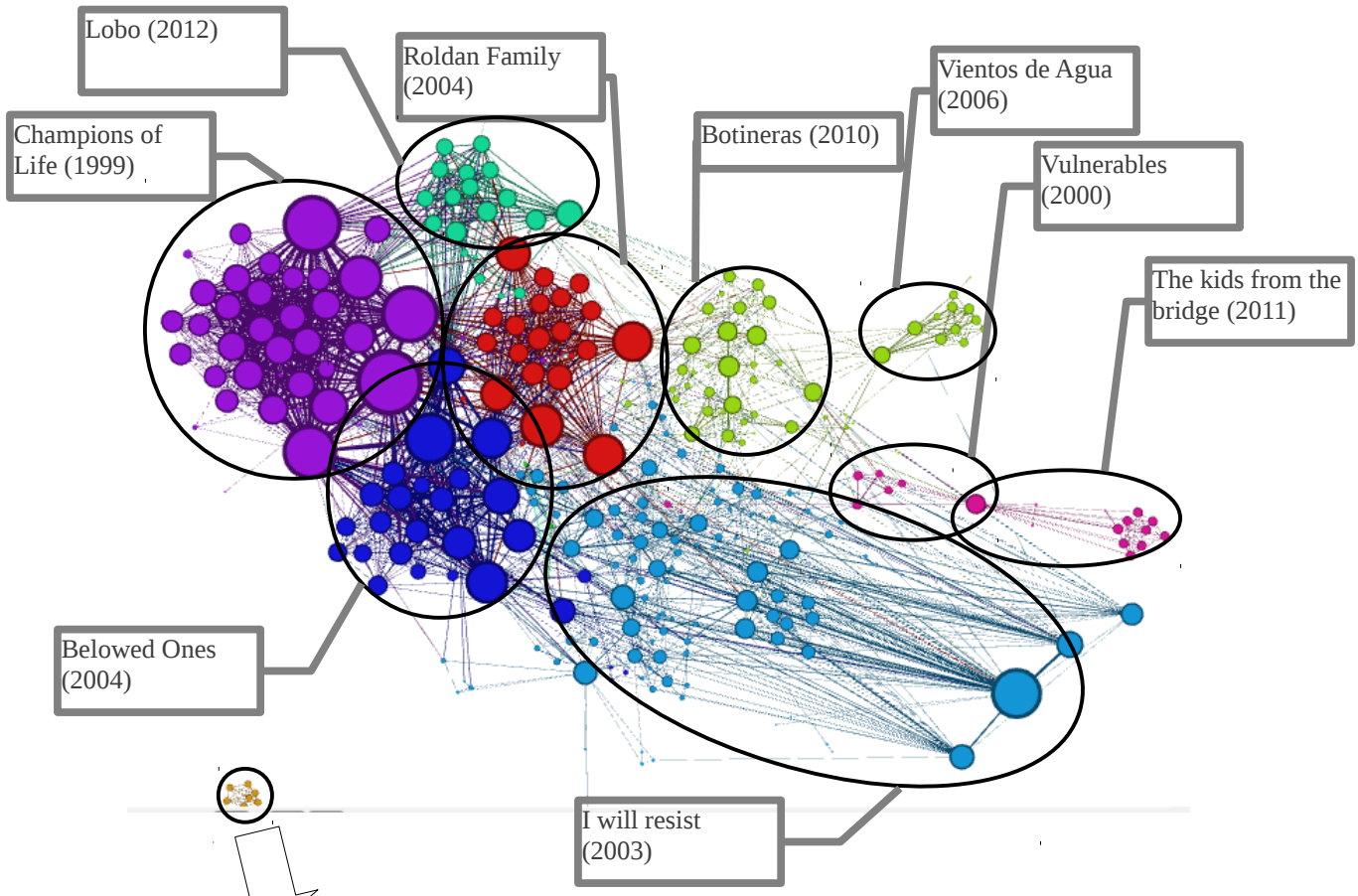
(Picture #17)

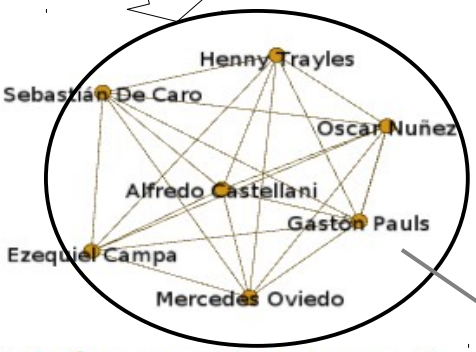


(Picture #18)

The red community (Picture 18) is the tv show “Los Roldan” (The Roldan's Family) [2]. Some actors (Luciano Castro, Facundo Espinosa) from the purple community were also part of the tv show “Los Roldan”. But they had stronger ties to the tv show “Champions of Life”.

The Blue community (Picture 18) has actors from the tv show “Son Amores” (Belowed Ones) [3].





This yellow Clique means that all the actors have worked together almost 6 times. They are disconnected from the big structure, that means that they have not worked almost 6 times with any other actor included in the network. This tv show is called "Todos Contra Juan" (Everybody against Juan, 2010) (Picture #19 with Wikipedia info page).

(Picture #19)

One may ask why the “Todos Contra Juan” clique is disconnected from the giant component. One reason may be because the this tv/show in contrast with the others was produced by a company called "Rosstoc", the only channel that accepted to play the tv/show was Channel 2 (the one with less ratings). The first from its two seasons from this tv/show was played in the Channel 2, the first season lose money and the Producing Company Rosstoc broke. (http://es.wikipedia.org/wiki/Rosstoc).

In contrast with the other Tv/shows, the Companies behind the tv/shows are more powerful and have more rating. There are actors that work more frequently by some Movies Companies than others, some of them are more expensive because they are more popular. It is supposed that these components are responsible for that clique disconnected from the big component.

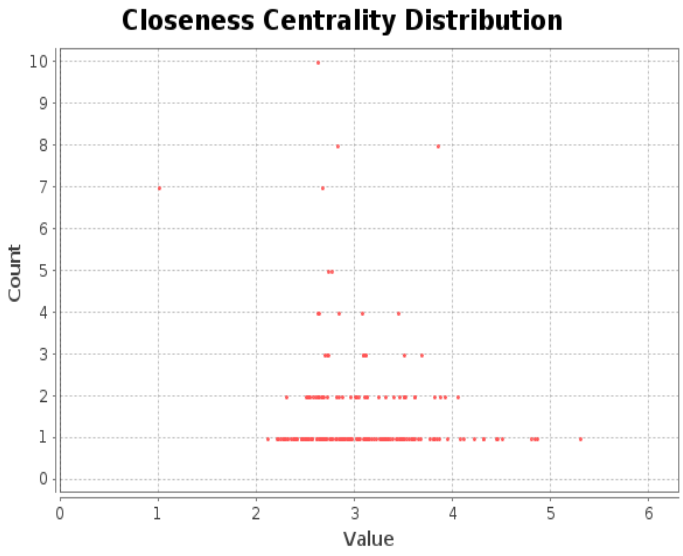
The light blue community are the actors from the tv show “Resistire” (I will resist) done in 2004 [4]. The pink community is actually two different tv shows with one actor in common that participated in both (Gustavo Garzón [12]). One tv show is called “Los pibes del puente” (The kids from the bridge) [5], the other is “Vulnerables” (Vulnerables) from 2000 [6].

TV Show Name	Produccion	Chanel	Year	
Son Amores	Pol-KA	13	2004	http://es.wikipedia.org/wiki/Son_amores
Campeones de la Vida	Pol-KA	13	1999	http://es.wikipedia.org/wiki/Campeones_de_la_vida
Botineras	Sebastian Ortega	11	2010	http://es.wikipedia.org/wiki/Botineras
Lobo	Pol-KA	13	2012	http://es.wikipedia.org/wiki/Lobo_%28telenovela%29
Los Roldan	Ideas Del Sur	9	2004	http://es.wikipedia.org/wiki/Los_Rold%C3%A1n
Vientos De Agua	Telecinco, Pol-ka, 100 Bares	13	2006	http://es.wikipedia.org/wiki/Vientos_de_agua
Vulnerables	Pol-KA	13	2000	http://es.wikipedia.org/wiki/Vulnerables
Los chicos del puente	INCAA	7	2011	es.wikipedia.org/wiki/Los_pibes_del_puente
Resistire	Telefe	13	2003	http://es.wikipedia.org/wiki/Resistir%C3%A9
Todos Contra Juan	Gaston Pauls	11	2010	http://es.wikipedia.org/wiki/Anexo:Todos_contra_Juan_2

(table #5)

e) Closeness Centrality

It may be interesting to know which is the distribution from the distances between each node and the others (Closeness Centrality).



Diameter: 8
Radius: 1
Average Path length: 3.0283151637364254
Number of shortest paths: 95214

(Picture #20)

The more central a node is, the lower its total distance to all other nodes. Closeness can be thought as a measure of how long it will take to spread information from *s* to all other nodes sequentially [8].

The 7 nodes that have a closeness of 1 are the disconnected clique of 7 nodes from the tv show “Todos Contra Juan” (All Against Juan).

The next actor, out of that clique is “Rafael Ferro” [7] (table #6).

Nodes	Id	Label	Degree	Modular...	PageRank	Eccentrici...	Closeness Centrality
Alfredo Castellani	4482.0	Alfredo Castellani	6	3	0.0031645570416003457	1.0	1.0
Sebastián De Caro	6570.0	Sebastián De Caro	6	3	0.0031645570416003457	1.0	1.0
Oscar Nuñez	460.0	Oscar Nuñez	6	3	0.0031645570416003457	1.0	1.0
Gastón Pauls	2143.0	Gastón Pauls	6	3	0.0031645570416003457	1.0	1.0
Ezequiel Campa	6751.0	Ezequiel Campa	6	3	0.0031645570416003457	1.0	1.0
Henny Trayles	2976.0	Henny Trayles	6	3	0.0031645570416003457	1.0	1.0
Mercedes Oviedo	6820.0	Mercedes Oviedo	6	3	0.0031645570416003457	1.0	1.0
Rafael Ferro	3833.0	Rafael Ferro	56	6	0.014258979854162955	4.0	2.107142857142857
Barbara Lombardo	3295.0	Barbara Lombardo	47	2	0.008924505761697 96	5.0	2.2077922077922 08

(Table #6)

Rafael Ferro is an actor that was in at least 22 different tv shows between 2001 and 2013, in which he was a relevant character (interacter more than 6 times in each tv show with some other character). This actor has high centrality, it may be because he is connected with many important acting figures (Picture #21).

(Picture #21)

f) Betweeness Centrality

The amount of pairs of individuals that need to go through an artist may give some idea of the importance of that movie character.

Nodes Edges Configuration Add node Add edge Search/Replace Import Spreadsheet Export table More actions Filter: Nodes

Nodes	Id	Label	Degree	Modular...	PageRank	Eccentrici...	Closeness Centrality	Betweenness Centrality
Rafael Ferro	3833.0	Rafael Ferro	56	6	0.014258979854162955	4.0	2.107142857142857	5148.019792421319
Gustavo Garzón	384.0	Gustavo Garzón	21	5	0.007569659517462154	6.0	2.8701298701298703	3945.184563754417
Tomás Fonzi	2319.0	Tomás Fonzi	45	2	0.009181026555423162	5.0	2.3084415584415585	3382.1423309141733
Raúl Rizzo	273.0	Raúl Rizzo	13	4	0.006248629271275195	6.0	2.811688311688312	2674.3937587683854

(Table #7)

Again, as in the Closeness Centrality, Rafael Ferro appears as an important figure in the actors network.

Betweenness centrality quantifies the number of times a node acts as a bridge along the shortest path between two other nodes [9].

The actor Rafael Ferro appears with the highers Betweenss Centrality, this may be taken as how well known is he in the overall Argentine actors community, how much experience he has acting with others.

3. Bibliography

[1] http://es.wikipedia.org/wiki/Campeones_de_la_vida

[2] http://es.wikipedia.org/wiki/Los_Rold%C3%A1n

[3] http://es.wikipedia.org/wiki/Son_amores

[4] http://es.wikipedia.org/wiki/Resistir%C3%A9

[5] http://es.wikipedia.org/wiki/Los_pibes_del_puente

[6] http://es.wikipedia.org/wiki/Vulnerables

[7] http://es.wikipedia.org/wiki/Rafael_Ferro

[8] M.E.J. Newman (2005), "A measure of betweenness centrality based on random walks", *Social Networks* **27**: 39–54

[9] http://en.wikipedia.org/wiki/Betweenness#Betweenness_centrality

[10] http://es.wikipedia.org/wiki/Facundo_Espinosa

[11] http://es.wikipedia.org/wiki/Gustavo_Garz%C3%B3n

4. Source Code

a) File: titulos.txt

Attached with the material.

b) Script: buscar_imdb.sh

```
#!/bin/bash
IFS=$'\n'
A=`cat titulos.txt`;
for j in $A; do
    Y=`echo $j | sed 's/ /+/g'`;
    wget "http://www.imdb.com/find?q=${Y}&s=tt" --quiet -O tmp.html
    ./buscar_imdb_1.pl tmp.html;
    echo -n $'\t'"$j";
    echo;
done;
```

c) Script: buscar_imdb_1.pl

```
#!/usr/bin/perl -w
use strict;
use LWP::UserAgent;
use LWP::Authen::Ntlm;
use Authen::NTLM;
use HTTP::Cookies;
use HTTP::Headers;
use MIME::Base64 qw(encode_base64);
use XML::Writer;
use XML::Parser;
use HTML::TreeBuilder;
use Digest::MD5 qw(md5_base64);
use Encode qw(encode_utf8);
use MIME::Base64;

if ($#ARGV+1 < 0) {
    print "buscar_imdb_1.pl <filename>\n";
    exit;
}
my $filename = $ARGV[0];
my $tree = HTML::TreeBuilder->new();
$tree->parse_file("$filename") or die ("No encontre $filename");
my @tablas = $tree->look_down(_tag=>'table', class => 'findList');
```

```
foreach my $t1 (@tablas) {
    my @tds = $t1->look_down('_tag' , 'td');
    my $i = 0;
    foreach my $t2 (@tds) {
            if ($i == 1) {
                    my $url = $t2->as_HTML;
                    $url =~ s/.*title\///g;
                    $url =~ s/\/\?ref.*//g;
                    print "$url";
            }

            $i++;
    }
}
```

d) Script: buscar_detalles.sh

```
#!/bin/bash
IFS=$'\n';
M=`cat res.txt`;

for i in $M; do
    echo -n "$i";
    wget "http://www.imdb.com/title/${i}/fullcredits?ref_=tt_ov_st_sm" --quiet -O tmp.html;
    wget "http://www.imdb.com/title/${i}/?ref_=ttfc_fc_tt" --quiet -O genre.html;
    ./buscar_detalles_2.pl genre.html;
    ./buscar_detalles_1.pl tmp.html;
    echo;
done;
```

e) Script: buscar_detalles_2.pl

```
#!/usr/bin/perl -w
use strict;
use LWP::UserAgent;
use LWP::Authen::Ntlm;
use Authen::NTLM;
use HTTP::Cookies;
use HTTP::Headers;
use MIME::Base64 qw(encode_base64);
use XML::Writer;
use XML::Parser;
use HTML::TreeBuilder;
use Digest::MD5 qw(md5_base64);
```

```
use Encode qw(encode_utf8);
use MIME::Base64;

sub trim($) {
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}

if ($#ARGV+1 < 0) {
    print "buscar_detalles_1.pl <filename>\n";
    exit;
}
my $filename = $ARGV[0];
my $tree = HTML::TreeBuilder->new();
$tree->parse_file("$filename") or die ("No encontre $filename");

my $typeshow = $tree->look_down(_tag=>'meta', property=>'og:type');
exit if ($typeshow->as_HTML =~ /video.tv_show/);

my $pais = "";
my $elanio;
my @tablas = $tree->look_down(_tag=>'div', class => 'txt-block');
foreach my $t1 (@tablas) {
    my @tds = $t1->look_down('_tag'=>'a');
    foreach my $tadeo (@tds) {
           if (defined($tadeo)) {
                  my $llo = $tadeo->as_HTML;
                  if ($llo =~ /country/) {
                         $llo =~ s/.*">//g;
                         $llo =~ s/<\/.*//g;
                         $pais = "$llo;$pais";
                  }
           }
    }
}
print "\t";
my @genre = $tree->look_down(_tag=>'span', itemprop=>'genre');
foreach my $g (@genre) {
    my $tt = $g->as_text;
    print ";$tt";
}
my @meme = $tree->look_down(_tag=>'meta', itemprop=>'datePublished');
```

```
if (!(defined($meme[0]))) {
    $elanio = "";
} else {
    $elanio = $meme[0]->as_HTML;
    $elanio =~ s/.*tent="//g;
    $elanio =~ s/".*//g;
}
print "\t$pais\t$elanio";
```

f) Script: buscar_detalles_1.pl

```
#!/usr/bin/perl -w
use strict;
use LWP::UserAgent;
use LWP::Authen::Ntlm;
use Authen::NTLM;
use HTTP::Cookies;
use HTTP::Headers;
use MIME::Base64 qw(encode_base64);
use XML::Writer;
use XML::Parser;
use HTML::TreeBuilder;
use Digest::MD5 qw(md5_base64);
use Encode qw(encode_utf8);
use MIME::Base64;

sub trim($) {
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}


if ($#ARGV+1 < 0) {
    print "buscar_detalles_1.pl <filename>\n";
    exit;
}
my $filename = $ARGV[0];
my $tree = HTML::TreeBuilder->new();
$tree->parse_file("$filename") or die ("No encontre $filename");
```

```
my $typeshow = $tree->look_down(_tag=>'meta', property=>'og:type');
exit if ($typeshow->as_HTML =~ /video.tv_show/);

my @title = $tree->look_down(_tag=>'div', class=> 'parent');
print "\t" . $title[0]->as_text;

my @tablas = $tree->look_down(_tag=>'table', class => 'simpleTable simpleCreditsTable');
print "\t";
foreach my $t1 (@tablas) {
    my @tds = $t1->look_down('_tag' , 'td');
    foreach my $tdir (@tds) {
            my $dirname = trim($tdir->as_text);
            next if ($dirname eq "");
            print ";$dirname";
    }
    last;
}
print "\t";
@tablas = $tree->look_down(_tag=>'table', class => 'cast_list');
foreach my $t1 (@tablas) {
    my @trs = $t1->look_down(_tag=>'td', class=> 'itemprop');
    foreach my $t2 (@trs) {
            my $actorname = trim($t2->as_text);
            my $ttinfo = trim($t2->as_HTML);
            $ttinfo =~ s/.*\/name\///g;
            $ttinfo =~ s/\/.*//g;
            #$actorname =~ s/.*">//;
            #$actorname =~ s/<\/.*//g;
            print ";$actorname ($ttinfo)";
    }
}
```

g) Script: actores_peli.sh

```
#!/bin/bash
IFS=$'\n';
M=`cat act.txt | sed 's/.*(//' | sed 's/).*//g'`;

for i in $M; do
    echo -n "$i";
    wget "http://www.imdb.com/name/${i}/" --quiet -O pelis/${i}.html;
    echo;
done;
```

h) Script: buscar_pelis_actor.pl

```
#!/usr/bin/perl -w
use strict;
use LWP::UserAgent;
use LWP::Authen::Ntlm;
use Authen::NTLM;
use HTTP::Cookies;
use HTTP::Headers;
use MIME::Base64 qw(encode_base64);
use XML::Writer;
use XML::Parser;
use HTML::TreeBuilder;
use Digest::MD5 qw(md5_base64);
use Encode qw(encode_utf8);
use MIME::Base64;

sub trim($) {
        my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}


if ($#ARGV+1 < 0) {
    print "buscar_directores.pl <filename>\n";
    exit;
}
my $filename = $ARGV[0];
my $tree = HTML::TreeBuilder->new();
$tree->parse_file("$filename") or die ("No encontre $filename");

my @tablas = $tree->look_down(_tag=>'div', class=>'filmo-category-section');
my @lasa = $tablas[0]->look_down(_tag=>'a');

my $tmpid = 0;
my %titulos = ();
my %capitulos = ();
foreach my $t1 (@lasa) {
    my $tmpi = $t1->as_HTML;
    if ($tmpi =~ /title/) {
            my ($pelid) = $tmpi =~ /.*_act_([0-9]*)".*/;
```

```
            next if (!(defined($pelid)));
            if (!(defined($titulos{"$pelid"}))) {
                    $titulos{"$pelid"} = $tmpi;
                    $capitulos{"$pelid"} = 1;
            } else {
                    $capitulos{"$pelid"}++;
            }
    }
}

foreach my $t1 (@lasa) {
    my $tmpi = $t1->as_HTML;
    if ($tmpi =~ /title/) {
            my ($pelid) = $tmpi =~ /.*_act_([0-9]*)".*/;
            next if (!(defined($pelid)));
            if ($capitulos{"$pelid"} == 1) {
                    print "$tmpi\n";
            } else {
                    next if ($titulos{"$pelid"} eq $tmpi);
                    my $titulo_nombre = $titulos{"$pelid"};
                    $titulo_nombre =~ s/.*[0-9]"//g;
                    $titulo_nombre =~ s/<\/a.*//g;
                    print "==[$tmpi]===[$titulo_nombre]\n";
            }
    }
}
```

i) Script: mat2.pl

```
#!/usr/bin/perl

open F, "per2.txt" or die ("not possible;");

my %actores = ();
my $cont = 0;
while (<F>) {
    chomp;
    my @actores_l = split(/;/);
    foreach my $a (@actores_l) {
            if (!(defined($actores{"$a"}))) {
                    $actores{"$a"} = $cont;
                    $cont = $cont + 1;
            }
    }
```

```
}


print "graph [
    directed 0
";

my %edges = ();
seek (F, 0,0) ;
while (<F>) {
    chomp;
    my @actores_l = split(/;/);
    for (my $i; $i < scalar(@actores_l); $i++) {
            for (my $j = ($i+1); $j < scalar(@actores_l); $j++) {
                    my $a = $actores_l[$i];
                    my $b = $actores_l[$j];
                    my $ejenombre;
                    if ($actores{"$a"} < $actores{"$b"}) {
                            $ejenombre = $actores{"$a"} . "," . $actores{"$b"};
                    } else {
                            $ejenombre = $actores{"$b"} . "," . $actores{"$a"};
                    }
                    if (!(defined($edges{"$ejenombre"}))) {
                            $edges{"$ejenombre"} = 1;
                    } else {
                            $edges{"$ejenombre"}++;
                    }
            }
    }
}

my @aparecen = ();
foreach my $k (keys %edges) {
    my ($a,$b) = split (/,/, $k);
    my $c = $edges{"$k"};
    next if ($c < 6);
    if (! ( grep { $_ eq "$a"} @aparecen)) {
            push (@aparecen, $a);
    }
    if (! ( grep { $_ eq "$b"} @aparecen)) {
            push (@aparecen, $b);
    }
}
```

```
foreach my $k (keys %actores) {
    if ( (grep({$_ eq $actores{"$k"} } @aparecen))) {
            print "\tnode [\n\tid " . $actores{"$k"} . "\n\tlabel \"$k\"\n]\n";
    }
}


foreach my $k (keys %edges) {
    my ($a,$b) = split (/,/, $k);
    my $c = $edges{"$k"};
    next if ($c < 6);
    print "edge [\n";
    print "\tsource $a\n\ttarget $b\n\tweight $c\n\t]\n";
}
print "\n]";
close F;
```