Ferran Pérez, U160396
Rosa Alós, U150080
P202

Assignment 1
Network generation and display

1. Introduction

In order to complete this assignment, we had to complete two different tasks. The first one consisted in splitting a metadata document into four separated csv's. After that, we had to create two graphs from these documents.

The first thing felt easy at the beginning, but as we dug deeper and deeper into the matter, we realized that it was far from simple. The main problems were the scalability and the control of duplicates and blank spaces. We started by checking every single field and attribute at every iteration, but that yielded poor performance, and therefore generating the documents lasted forever. To address this issue, we decided to introduce a class structure to contain the information related to movies and actors. This way, the original document only had to be read once, and all the documents were generated after the digital objects. In the same spirit, we also introduced functions that encapsulated repetitive behaviors, such as trimming the strings or intersecting lists. This may make the code more complex at first sight, but it got more intuitive to understand, and we think it's a change in the good direction.

We could talk this entire report about the code and the decisions we made while implementing it, but we think that it's more important to discuss about the second part of the assignment. In the end, this is introduction to data science, isn't it'?

2. Understanding the graphs

2.1. Movie-Movie graph

Fig. 1. The entire movie-movie graph

One thing that caught our attention when we first saw this graph was how many components it has. Our first ideas leaned to the believe that most of the movies will be connected on a large component, since we thought that in the end, actors and directors work in a lot of movies, and some actors have made dozens of films along his career, leading into more edges than the ones we were observing.

The simplest reason we could find about observing this unexpected pattern has to do with the fact that, in every movie, there's way more than three actors performing. What's the criteria used to decide which three actors get chosen into each movie? We may see films sharing more than ten actors without an edge between them (sequels for example) just because they changed the director and the three characters displayed on the database don't match. If our original csv had a list with at least the first ten main actors, we could see more edges between movies, leading to a larger connected component.

But how movies interact with each other? What shapes they produce? We will need to zoom into different parts of the graph to answer these questions.

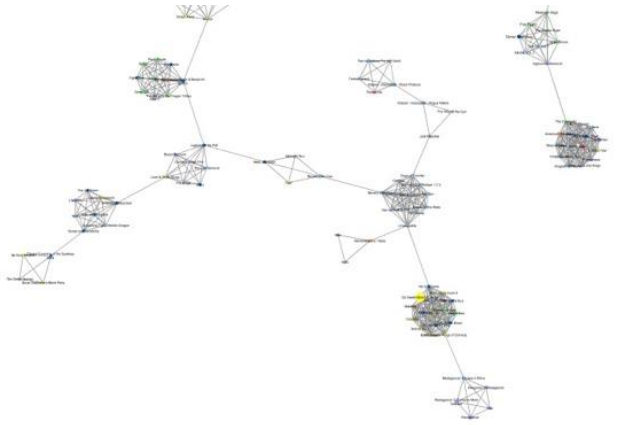

Fig. 2. Zoom of the most connected component

This picture serves to illustrate how the nodes in this graph behave. It belongs to the most connected component of the graph, although the behavior along all the major component follows the same pattern. We can see that the nodes group in cliques, and that is because all the movies directed by the same director are related among themselves. Between these cliques, there are some edges created by the 2 or more actors in common rule. These are less frequent, as we discussed above, and they are the ones that keep the component connected.

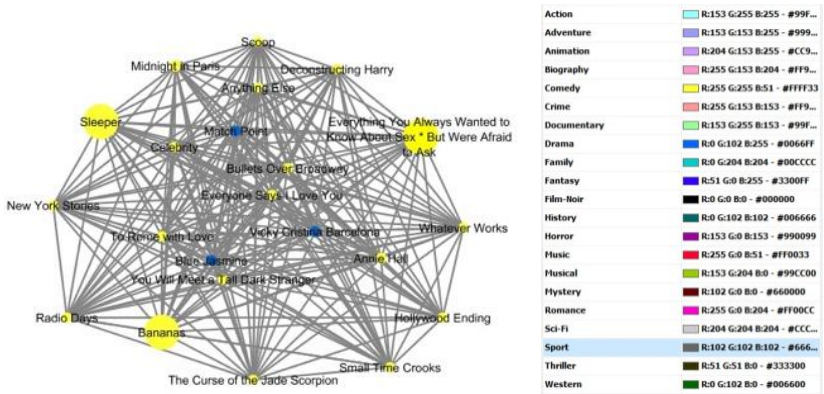

Fig. 3. Zoom of a particular clique

It's trivial that what determines the size of these cliques is how prolific a director is. Another thing we can notice from this last figure is that almost all the movies in cliques share a genre. That's because a director uses to produce the same kind of films. This clique, for example, is from Woody Allen. Almost all the movies Woody Allen has directed are related (don't forget we only use the first genre on the DB to determine the color) with the comedy. Following the same logic, although not all of them are this homogenous, the most dominant genres in every clique are close to each other. Action and Sci-Fi, or Action and Drama, are examples of this.

We can deduce that this graph has a physical limit, causing a saturation of edges a node can have. This happens because a director or an actor can only make a certain number of movies along his career, and therefore a node's degree won't escalate indefinitely as the graph grows larger.

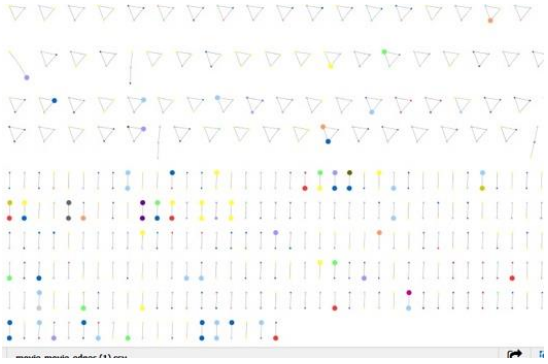

Fig. 3. Zoom of the less connected components

In the same lane, the less connected components in the graph are connected mostly by edges created by a director in common. Don't forget that these kinds of edges are a lot more frequent than the other ones, so it makes sense that the same proportion is maintained whether the director directed 3 movies or 20.

2.2. Actor-Actor graph

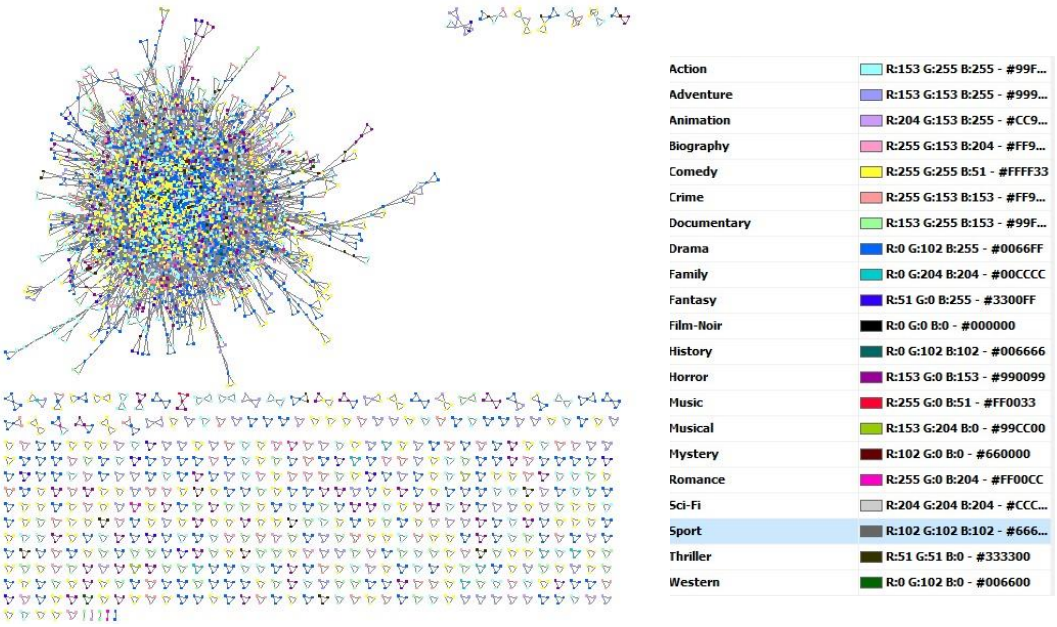

Fig. 4. The entire actor-actor graph

This one, on the other hand, is exactly as we expected it. There's a super component that contains almost all the actors, and some couples and trios created when the actors of a movie don't appear anymore on the DB after the first time. The small components all share the same genre (the genre of the only movie where they appear). However, the connected component is heterogeneous. One closer look may help understand how it behaves...

Fig. 5. The connected component

...or not. It looks like a mess! We can say that the nucleus is formed by the super Hollywood stars with an average degree close to 60. All of them are well-known actors. There is a bias caused for the fact that only the main characters of a certain movie get displayed in the DB, so although a secondary actor acted in dozens of movies, it won't get reflected on this graph. About the mixture of colors, we can only hypothesize that the genre of an actor is arbitrary.

As an example, someone could act in five Pirates of the Caribbean movies and appear as a comedy actor, and that happens because the genres in the database are not pondered. If we say that a film has action or drama or sci-fi as its main genre, but there's a component of comedy on each one, it will seem like an actor acting in all these movies is a comedy actor just because we count all the name appearances by the same amount.

In order to match this graph with the theory we've seen in class, it reminds us of a scale free network. We observe the large connected component, and if we check its node degree distribution plot in a log axis, although it doesn't describe a perfect straight line, it looks like one.

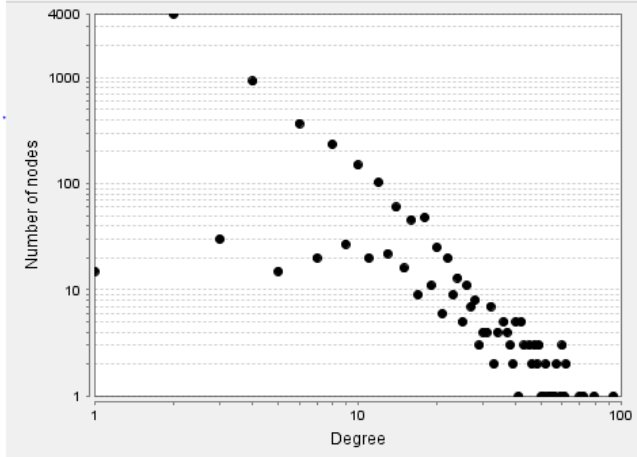

Fig. 6. Node degree distribution plot

Finishing, we could say that the nodes on this graph get saturated in the same way than the first one. One actor can only act in a certain number of graphs, so its degree has a physical limitation.

3. Conclusions

The first thing that we learned upon doing this assignment is that one graph is as good as the database he comes from. If the database does not display well all the characteristics of its data, it's probable that the graph will lead to incomplete (or wrong) conclusions.

It's challenging to look at a graph as complex as this one and deduce logical conclusions. It's the first time we had to do it without guidance, and we believe that practice will help us to understand these kinds of graphs in a more complex way. However, we are satisfied about what we've learned from this assignment, and we enjoyed unraveling it.

We hereby declare that, except for the code provided by the course instructors, all of our code, report, and figures were produced by ourselves.