UCLouvain

ÉCOLE POLYTECHNIQUE DE LOUVAIN

LINMA2472 - Algorithms in data science

Project 1: Co-occurrence Network

Authors : Heroufosse Gauthier
Degives Nicolas
Gandjeto Martine

2021 - 2022

Introduction

In this first project, the main objective is to try your hand at network construction. But what is a network? A network is a structure that represent a set of objects in which some pairs of the objects are in some sense "related". The objects correspond to mathematical abstractions called nodes (also called vertices or points) and each of the related pairs of vertices is called an edge.

The goal of the project was to build a co-occurrence network of a work. Our group chose to work about 21 Jump Street, a funny film released in 2012. This film is particularly suited to our purpose, as it has many different characters and scenes. After constructing the network, we had to analyze its multiple properties, and compare them with existing graph models. All our plots were made with the python library matplotlib.

Part 1: Construction of the co-occurrence Network

Before constructing the network we perform some pre-processing steps of the text (the movie script) data (Raghav, 2021). Our first step consisted in separating the movie script into different scenes. Given the format of our script, it was appropriate to cut the text at each tabulation (appearing as "/t" in the .txt file) to delimit the scenes. Next we make sure that all words in text have lower case. As we could not create a successful character recognition, we collected ourselves a list of the characters that appear in the movie from an online platform. In total we have 50 characters and 416 scenes.

To construct the network, we decided to define the co-occurrence of two characters when they appear in the same scene. After having identified the characters that appear in each scene, we create the co-occurrence matrix of dimension 50×50. The row and the column of the matrix are characters' name and the elements of the matrix are the number of time two different characters appear in a scene. We called this co-occurrence matrix weighted. We also construct an unweighted co-occurrence matrix where the elements are 0 (when the characters do not appear in the same scene) and 1 (when characters are in the same scene). We then use these matrices to construct the co-occurrence network.

	schmidt	jenko	molly	eric	dickson	walters	domingo	hardy	gordon	samuels
schmidt	127	176	54	68	18	12	20	2	10	6
jenko	176	117	40	68	18	6	18	2	2	2
molly	54	40	31	32	0	8	4	0	10	6
eric	68	68	32	44	4	6	4	0	2	0
dickson	18	18	0	4	11	0	0	0	0	0
walters	12	6	8	6	0	9	4	0	0	0
domingo	20	18	4	4	0	4	13	0	0	0
hardy	2	2	0	0	0	0	0	2	0	0
gordon	10	2	10	2	0	0	0	0	5	4
samuels	6	2	6	0	0	0	0	0	4	3

Figure 1: 10 first row/column of the co-occurrence matrix (weighted)

The graph 2 below displays the network that we obtained using the weighted co-occurrence matrix.The nodes (characters) are placed by importance and their size represents the number of times they are quoted in the script. The thickness of the edges reflects the strength of the link between the two characters (the number of scenes where they both appear).

Part 2: Analyzing the graph

In the different courses, we have seen several ways to analyze the properties of networks. We can distinguish 3 levels of abstraction:

1. Element-level analysis, where methods try to identify the most important nodes of the network;
2. Group-level analysis, that involves methods for finding and clustering cohesive groups of nodes in the network;
3. Network-level analysis, that focuses on topological properties of networks as a whole.

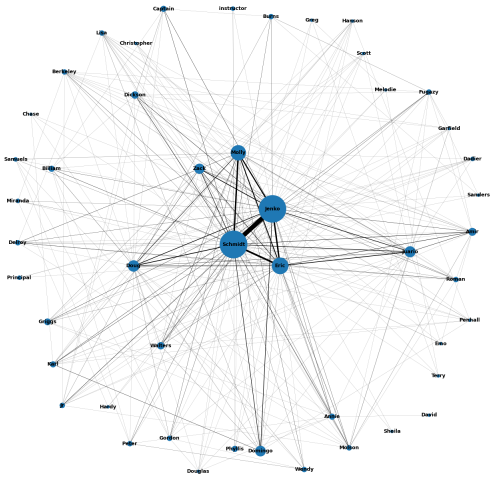

Figure 2: Visualization of the co-occurrence network (weighted)

We will present some relevant properties for the first two levels. For the analysis at the network level, we can mention for example the clustering coefficient which presents appropriate information. It is also known as transitivity, and in everyday language we translate it as "my friend's friend is also my friend". This coefficient returns the probability of 3 nodes forming a triangle.

2.1 Degree assortativity

The Assortativity coefficient (Hiroki, 2019) is analogous to the Pearson correlation coefficient that assesses the association between two continuous variables. In fact, it measures the correlation between the characteristics (e.g. degree) of every pair of nodes that are connected and its values range between -1 and 1. It reflects the tendency for nodes to connect with nodes with dissimilar properties within a network.

A large positive value means that connected nodes very much tend to share similar properties; a large negative value means that connected nodes tend to possess very different properties. In our weighted network, assortativity measure the degree property.

In our case, we get a negative value of **-0.276**. This means that our network of co-occurrence reveals dissimilarity of the degree of the characters. In fact, we could see from the figure 2 that there are a few characters who connect with a lot of people, when the rest tend to only have a small number of relationships. It represents an heterogeneous network.

2.2 Community detection using Louvain's Algorithm

The second property we computed is network clustering. Network clustering is an organizing process whose goal is to bring together similar nodes; the result is a partition of the network into a set of communities. The effectiveness of network clustering can be measured by modularity. Modularity is a measure of the structure of networks or graphs that measures the strength of the division of a network into communities (Wikipedia, 2021) .

Here, we used the 'Louvain algorithm' as a tool to maximize the modularity of our network (as discussed in Lecture 2). It starts with individual nodes as a community, and for each node in turn, it tries to merge it with another community if this increases modularity. Because it chooses the starting node randomly, one of its strengths is the randomness of this method. This also means that you can get multiple results from the same network partitioning code. To confirm that your partitioning is relevant, it is highly recommended to do a few iterations. If the results are the same, you can certainly say that your partitioning is relevant. This process is called robustness of your method: the more identical your results are, the more robust your method is, and

the Louvain algorithm is known as one of the most robust clustering methods (Blondel et al., 2008).

To realize our partition graph, we use the community package on Python, which directly executes the Louvain algorithm. As we said above, the Louvain algorithm needs several iterations to be considered reliable. We arbitrarily chose to do 100 iterations and use the average partition as our main graph. As you can see in the graph below, we found a partition of 4 communities as the final result.

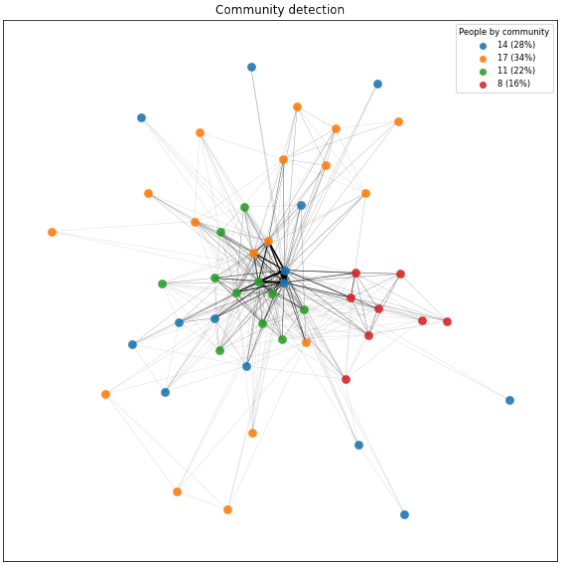

Figure 3: Community detection using Louvain Algorithm

2.3 K-core decomposition

In general, the goal of K-core decomposition is to identify groups of nodes which are tightly connected. This means that in our case, we want to determine characters from the co-occurrence network who are highly interconnected. To perform k-core decomposition, several algorithms exist (Malliaros et al., 2019). For our analysis we have implemented the most commonly used algorithm which is a pruning process that recursively remove the nodes that have degrees less than k (as described in the course lecture). The k-core is typically the remaining nodes.

We created a function to determine the core for different values of k by considering the unweighted network. We proceed in two main steps:

- Initialization: here we determine the neighbors of each node as well as their degree. We define a vector 'pruned' to track nodes that are removed at each iteration. We also initialize the counter of pruned nodes to 0.
- Iteration: Here our algorithm visits each node in the network and remove them if they have a degree that is less than k

We complement these steps by a third one where we assign to each node its core number. Our defined function requires one main input which is the graph and returns two outputs (on one side the shells for different values of k and on the other the mapping between each node and its core number).

By applying the function to the unweighted co-occurrence network, we are able to identify the following shells:

- 1-shell: empty
- 2-shell: Sheila, Janitor, Instructor
- 3-shell: David, Hardy
- 4-shell: Christopher, Accompanist, Miranda

- 5-shell: Melodie, Phyllis, Greg, Terry
- 6-shell: empty
- 7-shell: Scott, Sanders
- 8-shell: Doug, Dickson, Domingo, Zack, Lisa, Samuels, Griggs, Jenko, Molly, Douglas, Dadier, Delroy, Jr, Principal, Emo, Amir, Gordon, Berkeley, Garfield, Walters, Chase, Hanson, Schmidt, Burns, Molson, Karl, Wendy, Fugazy, Eric, Annie, Peter, Roman, Juario, Billiam, Penhall, Captain

We are also able to visualize the k-cores (highlighted with different colors) in the figure 4. Overall we identified a group 36 characters who are highly interconnected and who have little co-occurrence relationship with others. This reveals that there exists some hierarchy among characters in the movie.

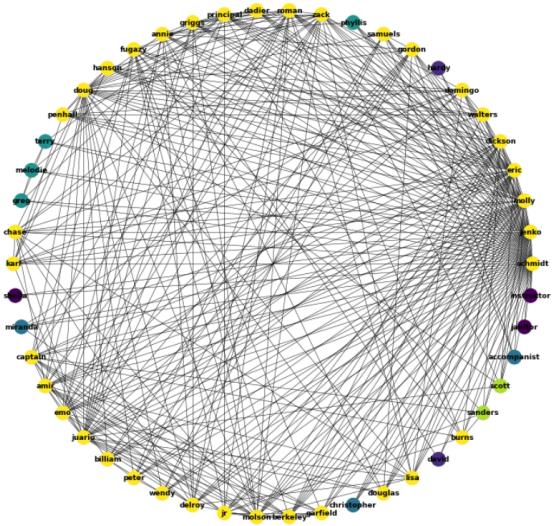

Figure 4: Visualization of k-cores

Part 3: Comparison with the Babarasi-Albert model

The next section of our report tries to compare our network with a well-known scale-free model, the Babarasi-Albert network. Its authors suggested that several natural and human-made systems, such as Internet, citation network and some social networks are thought to be approximately scale-free networks. A scale-free network is a network whose degree distribution follows a power law, at least asymptotically. That is, the fraction P (k) of nodes in the network having k connections to other nodes goes for large values of k as $P(k) = ck^{-\gamma}$, where γ is a parameter whose value is typically in the range $[2, 3]$.

A key concept of such network is the preferential attachment property. It means that the more connected a node is, the more likely it is to receive new links. Nodes with higher degree have stronger ability to grab links added to the network. As we can see, it seems quite similar in our network, if you compare the assortativity coefficient of the two networks. The Barabasi-Albert graph gets a **-0.139** and our is **-0.276**. Both networks reveals dissimilarity of the degree of the characters. To go further, we could compare the degree distribution or the clustering coefficient.

Part 4: Influence Maximization problem

This part deals with an influence maximization problem. An important rumor appears and we try to predict from which character this one will spread the fastest in all the network. In order to simulate this propagation, we use the greedy algorithm seen in class in order to detect the characters susceptible to spread the rumor the fastest and the independent cascade model to estimate the propagation from the first characters made aware of this rumor.

4.1 Greedy algorithm

The greedy algorithm chooses the optimal solution that presents itself at each moment until it has completely solved the problem. It does not care about the previous or the next step.

The algorithm starts by choosing the best initiator for the rumor and add it in the seeds nodes. It then chooses the second one which has the best marginal gain compared to the first one and so on until it has the set of k seed nodes. The greedy algorithm thus makes locally optimal choices in the hope that this is also the globally optimal solution.

The advantage of this algorithm is that it only has to calculate the propagation of $\sum_{i=0}^{k}(n-i) \approx kn$ nodes. But this implies that the seeds nodes found are only an approximation of the maximum influence problem because it only considers the individual propagation of k starting nodes and not the combination of them. However, the algorithm is theoretically supposed to choose a seed set whose spread will be at least 63 percent of the spread of the optimal seed set.

We have run this algorithm on our network to identify the three best seed nodes (k=five percent of the nodes). We set to 0.1 the probability that the node activate an edge and thus spreads to the neighbor corresponding to this edge. The results are unsurprisingly Schmidt, Jenko and Eric. These are obviously the main characters in the story and therefore corresponding to the nodes with the most edges in the case of many movies.

4.2 Independent cascade model

The spread process model used is the Independent Cascade model.

This model predict based on the seeds nodes what could be the diffusion of the rumors in the network. For each edge, there is a weight of $p_{u,v} \in [0, 1]$ which can be seen as the probability to be 'infected'. At each time step, the infected nodes can infect the neighboring nodes with a probability of p. A particularity of this model is that a node infected at time t can only infect its susceptible neighbors at time t+1. After that time, it can no longer transmit the rumor. We also assume that an individual who is aware of the rumor does not forget it and therefore remains among the 'infected'.

In order to appreciate the contribution of the greedy algorithm. We have compared different independent cascade model. A first one uses the starting nodes found by the greedy algorithm (red curve) and eight other simulations start from random nodes.

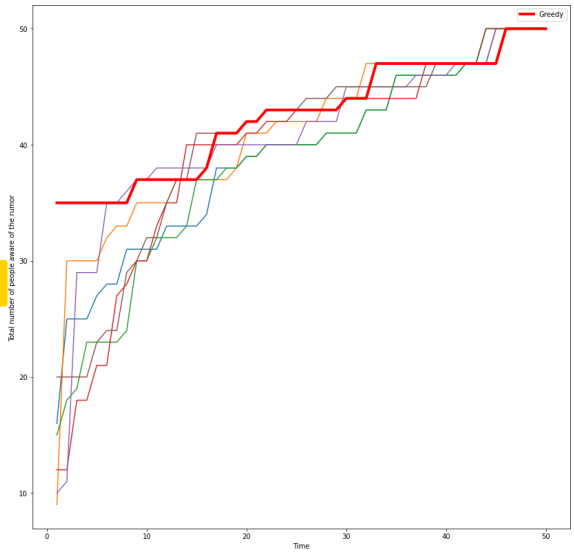

Figure 5: Cascade model with random initiator

We notice that the greedy algorithm works quite well because it is the highest curve at time t=1, which means that the three starting nodes are well chosen for a fast propagation start.

Conclusion

In this first project, we had to analyze network properties at element-level analysis and group-level analysis. For the element-level analysis, we found that our network present a dissimilarity of degree between characters that are linked. We compared our networks with a typical scale-free model (Babarasi-Albert model) and we noticed common property between both graphs. Then we found via the greedy algorithms the characters that are the most reliable for information propagation. Concerning group-level analysis, we used the Louvain-Algorithm in order to maximize the modularity of our network. We found that our network could be organised in 4 different communities.

Although this preliminary analysis reveals interesting features of co-occurrence network we could have better insight by using others measures such as degree distribution, clustering coefficient and the transitivity.

Bibliography

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *2008*(10), P10008.
URL https://doi.org/10.1088/1742-5468/2008/10/p10008

Hiroki, S. (2019). Introduction to the modeling and analysis of complex systems.
URL https://math.libretexts.org/Bookshelves/Scientific_Computing_Simulations_and_Modeling/Book

Malliaros, F. D., Giatsidis, C., Papadopoulos, A. N., & Vazirgiannis, M. (2019). The core decomposition of networks: theory, algorithms and applications. *The VLDB Journal*, *29*, 61–92.

Raghav, A. (2021). Must known techniques for text preprocessing in nlp.
URL https://www.analyticsvidhya.com/blog/2021/06/must-known-techniques-for-text-preprocessing-i

Wikipedia (2021). Modularity (networks) — Wikipedia, the free encyclopedia. [Online; accessed 20-October-2021].
URL https://en.wikipedia.org/wiki/Modularity_(networks)

Other useful sources

- Network properties
- Geek Week (2018). Barabasi Albert Graph(for Scale Free Models)
- Github:IM GreedyCELF
- Github: Louvain Community Detection
- Networkx