Network Science - A.A. 2021/22

Amazon Co-purchasing Network

Fontana Francesco 2026924, Giorgetti Sabrina 2013375,
Latora Federica 2019043, Ziliotto Filippo 2017425

February 20, 2022

Abstract

Amazon is one of the world's biggest online marketplaces. This work aims to analyze its co-purchasing network data and find some useful business related insights. Since we have tons of data, one of the main problems was efficiency and optimization, so we selected a subset category called *Movies & TV*. The main points of the analysis were: degree, assortativity, centrality measures, network robustness and community detection algorithms. As a final goal we implemented some *link prediction* algorithms in which, given a certain product, it is possible to predict which other items to suggest to the customer. This was done either with and without exploiting network properties such as Jaccard coefficient, resource allocation, preferential attachment and Node2Vec. Results show how these properties and the related area of research are of great importance to companies if they aim to increase revenue, since targeted recommendations of co-purchased products can effectively increase the average spend of customers.

1 Introduction

Over the last few years the phenomenon of online shopping has grown tremendously and more and more people are turning to websites for their purchases. Amazon is one of the leading companies in the e-commerce sector and this is why for this project we have chosen to focus on the Amazon co-purchasing network. This network allow us to study the feature of *"Customers Who Bought This Item Also Bought"*, which is of main interest for recommendation systems and in general to understand the market related to a product. This area of research is of great importance to companies since they aim to increase revenue and targeted recommendations of co-purchased products can increase the average spend of customers. The Amazon review dataset 2018 was built collecting data directly from the Amazon website and it includes reviews, metadata and links for 15.5 million products within the period May 1996 - Oct 2018. The data have been grouped into major categories, given both the huge amount of products and the dimensions of the files. Since the amount of meta-data available about Amazon products is huge, we decide to analyze only those products that belong to a certain category. To create the associated co-purchasing network and perform a proper study we have indeed decided to focus on the *"Movies and TV"* category only. The complete datasets can be found at

the following link: `http://deepyeti.ucsd.edu/jianmo/amazon/index.html` [1]. The analysis of the network was conducted within a Google Colab notebook and with Python as programming language. For the visualization of graphs and networks instead, the software Gephi is used.[1]

2 Creation of the co-purchasing network

After gathering the dataset, the first step was to process the initial file in order to extract only the useful information. In general the metadata for each product contains a variety of information such as the description, the feature, the title, the price and so on. In our case for the *"Movies and TV"* metadata we have filtered the dataset and selected only the attributes for each entries which are showed in Table 1.

ID	Numerical identification code of the product (e.g. 0000031852)
Title	Name of the product
Also buy	List of ID codes of the products bought by the same customer
Also view	List of ID codes of viewed product by the same customer
Price	Price of the object
Rank	Amazon ranking
Brand	Name of the company
Categories	List of tags associated to the product

Table 1: List of attributes for each product

After that, we discarded and removed all those products where some of the attributes' values were missing (i.e. if there were void entries or some NaN values): in the end we still have more than 30 000 entries. From the representation shown in Figure 1, it is possible to notice the quite huge dimension of the network: specifically it presents 33 594 nodes and 401 714 edges.

We proceed to create the co-purchasing network unwrapping the *"also buy"* attribute list for each product: this can be easily done thanks to the Python function `explode()`, which transforms each element of the list to a row and replicates the index values (in our case, the ID of the product). The resulted edge list is a Pandas DataFrame composed by two columns called `from_ID` and `to_ID`, which identify the ID codes of pairs of co-purchased products. The edge list file has more than 400 000 entries which will be indeed the edges of our graph (Fig. 2).

Thanks to the Python package *NetworkX*, which provides specific tools for the study of networks, we created the graph of the co-purchasing network from the edge list DataFrame. In the next sections we present the study of the features of the network and the respective results obtained.

[1]All the code for the project can be found at: `https://colab.research.google.com/drive/1-QDX1Lcjzn--oDuQgzSPfWDRdW5zRUzI?usp=sharing`

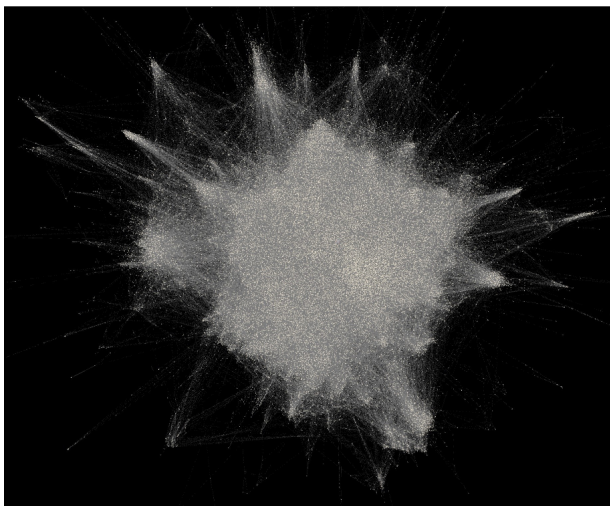

Figure 1: Entire network representation obtained with Gephi

	from_ID	to_ID
0	0000143588	B00N27ID1G
1	0000143588	B000NR7ROM
2	0000143588	B000AOEPMA
3	0000143588	B000UMP2VK
4	0000143588	B002TCRQ68
...	...	...
401709	B01HJ1INB0	B00DPMFE6U
401710	B01HJ1INB0	B00507ZQNO
401711	B01HJ1INB0	B000UPSKO8
401712	B01HJ6R77G	B0002F6BFG
401713	B01HJ6R77G	B01180FCR6

401714 rows × 2 columns

Figure 2: Pandas DataFrame containing the edge list

3 Analysis

3.1 Adjacency Matrix

One of the easiest ways to visualize a network from a mathematical point of view is through the *adjacency matrix* $A = [a_{ij}]$ [2]. It has rows and columns labeled by graph vertices, with a 1 or 0 in position a_{ij} according to whether node i and j are connected or not (Fig. 3). On a first view, it seems to present a lot of elements on the principal diagonal equal to one and consequently we might conclude that our network has a large number of self-loops. However, this is due to the fact that the matrix has quite huge dimensions (33 594 x 33 594) and consequently its entire representation is not so clear.

Indeed in the reality, the self-loops are only 464. This can be confirmed by looking at some zoomed versions of the matrix, where we can see that the majority of the diagonal elements are equal to 0, while there are many values around it equal to 1 (Fig. 4).

Moreover this matrix is asymmetric, so the graph is directed (i.e. the edges have a privileged direction).

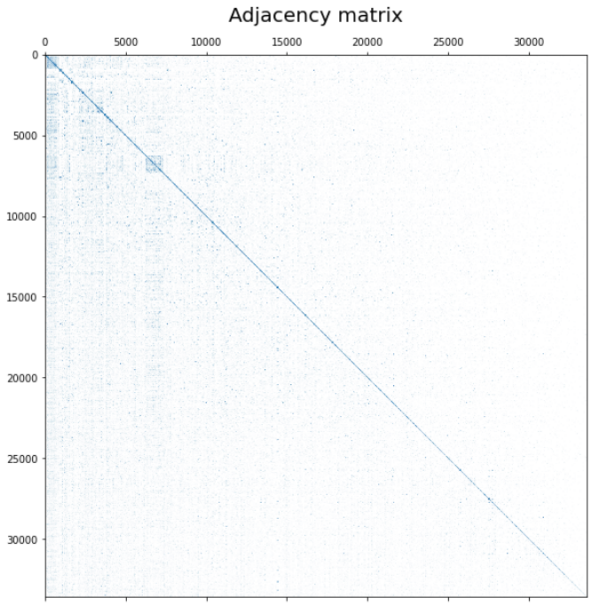

Figure 3: Adjacency matrix of the network

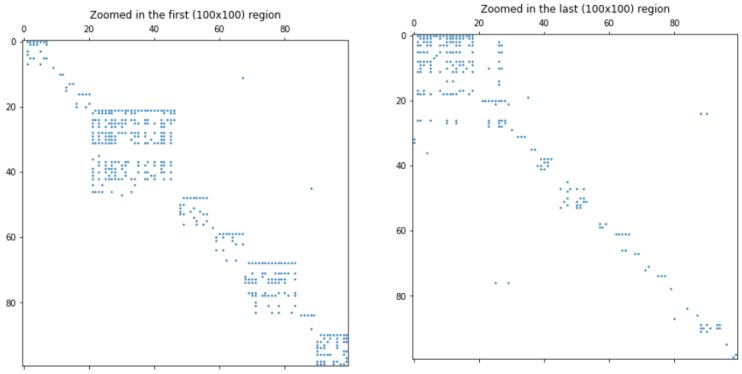

Figure 4: Some zoomed portions of the adjacency matrix

3.2 Network Degree

An interesting network metric is the *network degree*, which is the total number of links connected to a particular node. In our case, the plots in Figure 5 show that there are a large number of nodes with small degree compared to the number of nodes with high degree.

At this point, the *degree distribution* is studied in order to generally understand the behaviour of the network. However since this network is directed, we must distinguish

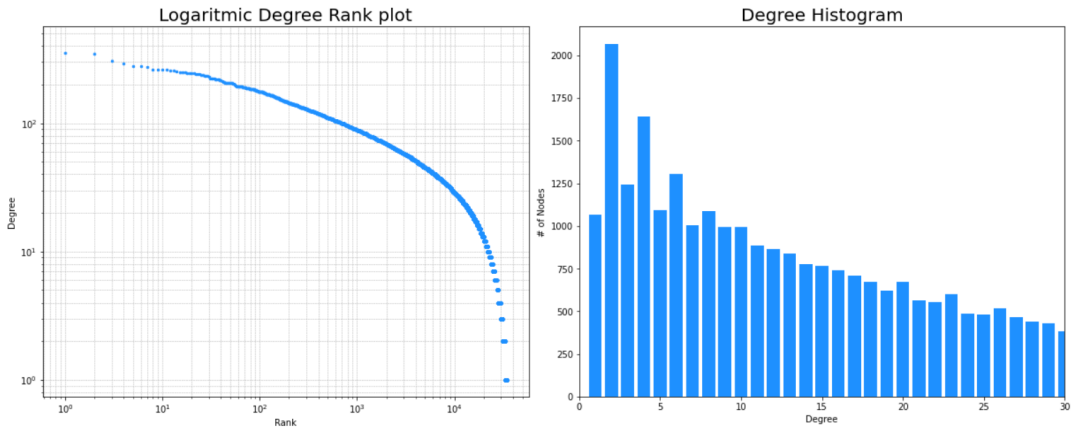

Figure 5: Network degree-rank plot and degree histogram

between in-degree k^{in}, which is the number of entering links, and out-degree k^{out}, which is instead the number of exiting links [2]. The main parameters of the two distributions are reported in Table 2.

	IN degree	**OUT degree**
Average degree $\langle k \rangle$	112.70	31.05
Second moment $\langle k^2 \rangle$	5425.83	312.90
Third moment $\langle k^3 \rangle$	265852.36	82.34
k_{max}	395	62
k_{min}	0	1
γ	3.21	11.38

Table 2: Main network parameters

For the *in degree distribution*, the power-law exponent γ is slightly higher than 3 and consequently the distribution does not follow perfectly a scale-free behavior ($2 \leq \gamma \leq 3$). However by looking at the divergence of the moments, we can see that they tend to explode as the order increases. This behaviour typically is verified in a scale-free network.

From the Probability density function (PDF) and the Complementary cumulative density function (CCDF) reported in Figure 6, the distribution seems to follow the power-law $p_k = Ck^{-\gamma}$ quite faithfully (the coefficient γ has been calculated via linear interpolation by discarding samples in the saturation region, i.e. samples with degree lower then $k_{min} = 105$). Moreover, the distribution is heavy-tailed and this is due to the presence of hubs, i.e. nodes with high degree.

For the *out degree distribution* instead, the γ coefficient is significantly higher than 3 (it has been calculated with $k_{min} = 52$), consequently the distribution holds a random behavior. This property can also be seen through the higher order momenta, because they remain finite and do not explode as their degree increases. Indeed it can be notice from Figure 7 that the distribution is not heavy-tailed, thus there is not a relevant amount of hubs.

The presence of more high degree nodes in the in-degree distribution can be seen by comparing the two logarithmic CCDF plots. Indeed for the in-degree distribution, the corresponding CCDF decreases more slowly than the one of the out-degree distribution.

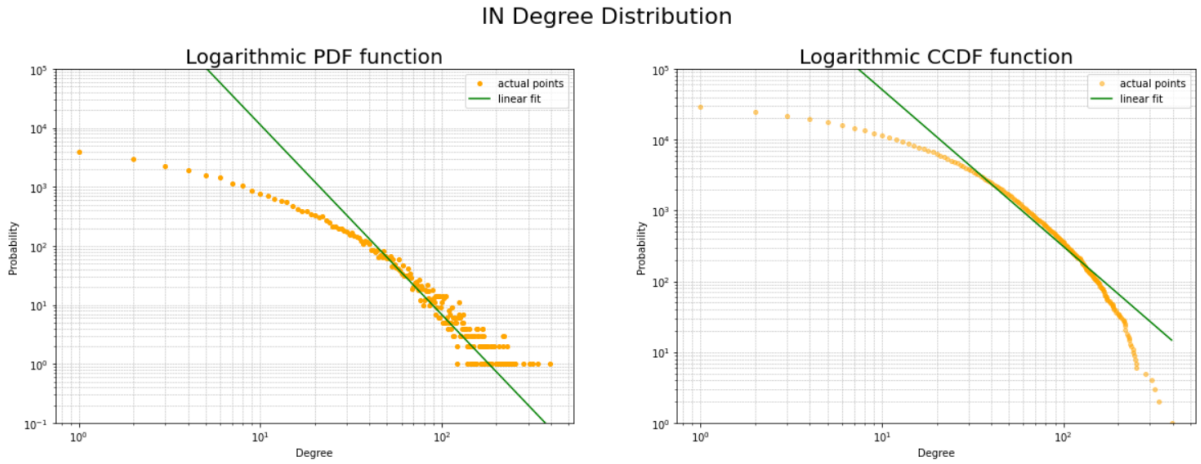

Figure 6: IN degree distribution: Logarithmic PDF and CCDF

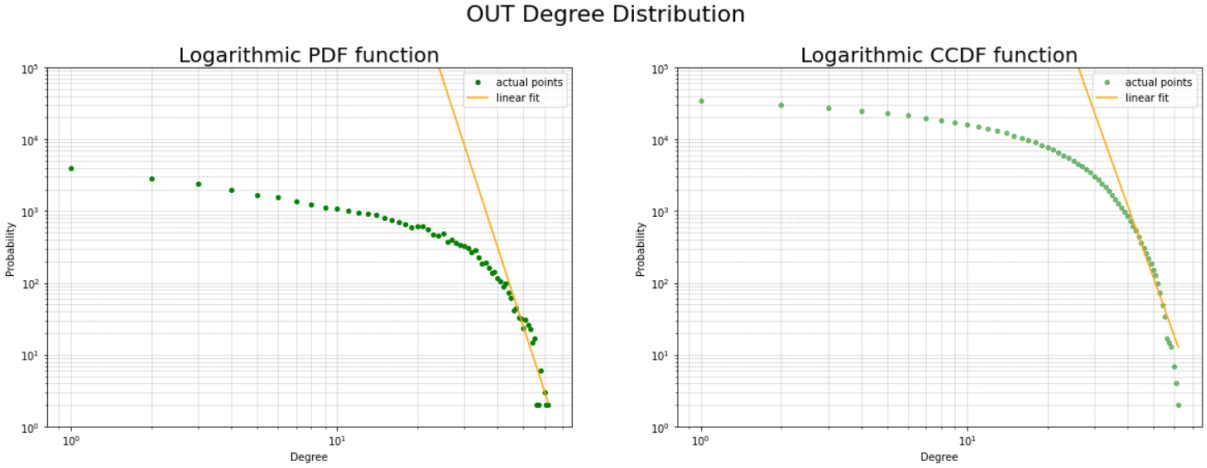

Figure 7: OUT degree distribution: Logarithmic PDF and CCDF

3.3 Assortativity

Assortativity can be defined as the tendency of nodes to prefer a connection with similar nodes with respect to dissimilar one, in particular high degree (low degree) nodes create links with other high degree (low degree) nodes.

The measure of assortativity can be evaluated through the *degree correlation matrix* E_{k_1,k_2}, in which we expect that nodes of comparable degree tend to link to each other (i.e. higher value along the diagonal). Since the matrix E_{k_1,k_2} contains the complete information about the degree correlations characterizing a particular network, it is difficult to interpret its content and to extract information from the visual inspection of this matrix. For this reason, a more compact way to detect degree correlations is performed by inspecting the degrees of the neighbouring nodes with a *degree correlation function*. We define the average degree of nearest neighbors $k_{nn}(k)$ of a node with degree k and study its trend with a linear fit. Indeed, the degree correlation function can be approximated with the expression $ln(k_{nn}) = \mu \cdot ln(k_i)$, where the sign of the correlation exponent μ determined the nature of degree correlations [3].

For a directed network we need to consider both in and out degrees of the nodes. The

results of the assortativity are shown in the Figure 8, from which it is possible to see that the network is assortative. Indeed for this network, all the values of the coefficients μ are positive (Tab. 3).

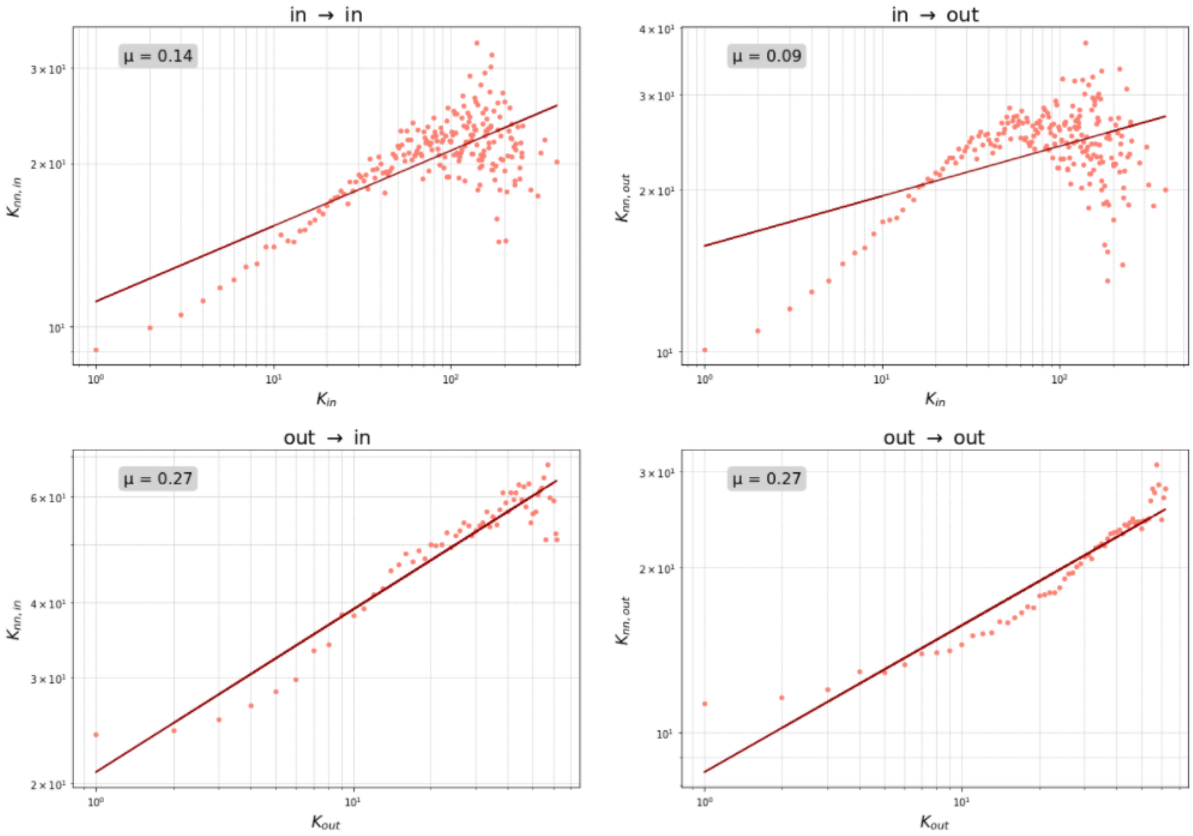

Figure 8: Assortativity

$\boldsymbol{\mu}$	in	out
in	0.14	0.09
out	0.27	0.27

Table 3: Assortativity parameters

An alternative parameter to measure the assortativity is the the *assortativity coefficient*, which is the Pearson correlation coefficient of degree between pairs of linked nodes. Notice that when it comes to directed networks we can actually measure four different type of coefficients associated with the in-degree and out-degree [4]:

- $r_{in,in}$: measures tendency of having a directed edge (i,j) such that $k_{in,i} = k_{in,j}$;
- $r_{in,out}$: measures tendency of having a directed edge (i,j) such that $k_{in,i} = k_{out,j}$;
- $r_{out,in}$: measures tendency of having a directed edge (i,j) such that $k_{out,i} = k_{in,j}$;
- $r_{out,out}$: measures tendency of having a directed edge (i,j) such that $k_{out,i} = k_{out,j}$.

For this network, the values of the coefficients r are reported in Table 4. Also in this case, all the coefficients are positive. We can conclude that our network is assortative.

$r_{in,in}$	$r_{in,out}$	$r_{out,in}$	$r_{out,out}$
0.13	0.1	0.21	0.28

Table 4: Assortativity coefficients

3.4 Authorities & Hubs

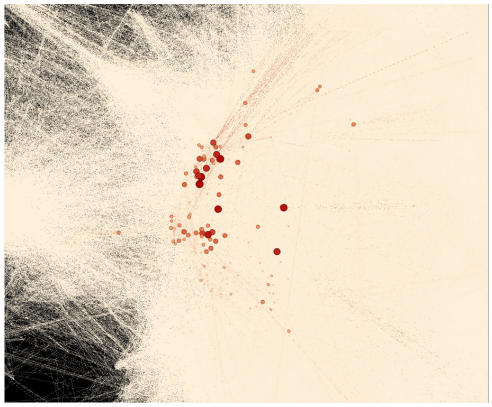

Figure 9: Authorities visualization. We see how the most important authorities are very close to each other compared the whole graph.

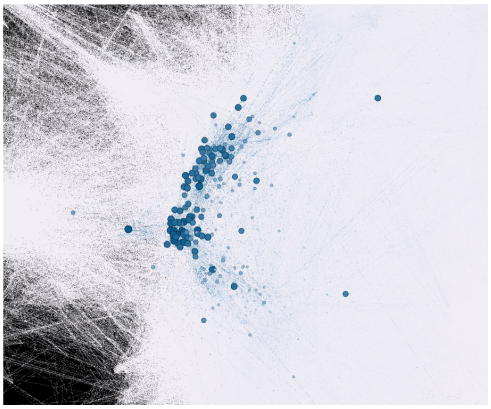

Figure 10: Hubs visualization. It seems that many of the biggest authorities (that are close to each other) correspond to the Hubs.

As we know we call *Authorities* nodes that contain useful information, or having a high number of edges pointing to them, while *Hubs* are trustworthy nodes, or nodes that link to many authorities. Both are related to the *HITS* algorithm, a link analysis algorithm that rates Web pages, developed by Jon Kleinberg[5].
Our Amazon network is composed by nodes that represent different items which are connected together by a co-purchasing choice by the customer. So, in other words a 'good' hub represents a item that points to many other co-purchased items, while a 'good' authority represents an item that is linked by many different hubs.
We performed an analysis through the *HITS* algorithm with the *NetworkX* library to find and compare the most relevant Hubs and Authorities, finally checking for items which were both Hubs & Authorities.
The results are shown comparing the in the Fig. 9 & Fig. 10. Visually the network suggests that many of the biggest hubs are also the biggest authorities and this is confirmed comparing the two associated dataframes. There are 34 common items in the first 100 scores for authorities and hubs. This tells us that 1/3 of the most important item nodes are both pointing to many 'trustworthy' nodes and having high number of incoming connections.This means that people tend to co-purchase the same item even if they initially bought a different one. Probably, it's due to the fact that the most sold products are the ones that people 'trust' the most based on other people shopping reviews. To confirm this hypothesis we compared authorities and hubs with the rank sale of the nodes and we saw

a positive correlation between the two. This correlation is highly visible for authorities while for hubs it seems less relevant (Fig. 12).
A final consideration to be added is that (as expected) the highest scores for both authorities and hubs are related usually to complete DVD series collection items (e.g. the highest authority which is also the 5^{th} is *'Season 1 Cheyenne'* (a tv-series of the 60's).

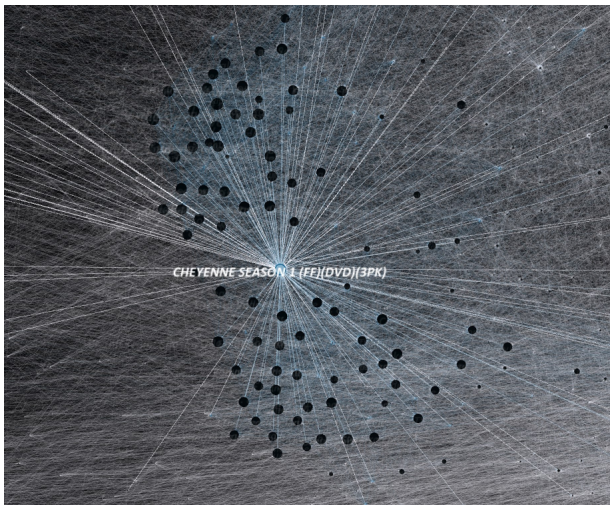

Figure 11: Biggest authority in the network and 5^{th} biggest hub. We see that the reason it's both authority and hub is the very high number of connections to other biggest hubs (blue nodes). It corresponds to the *SEASON 1 CHYENNE*.

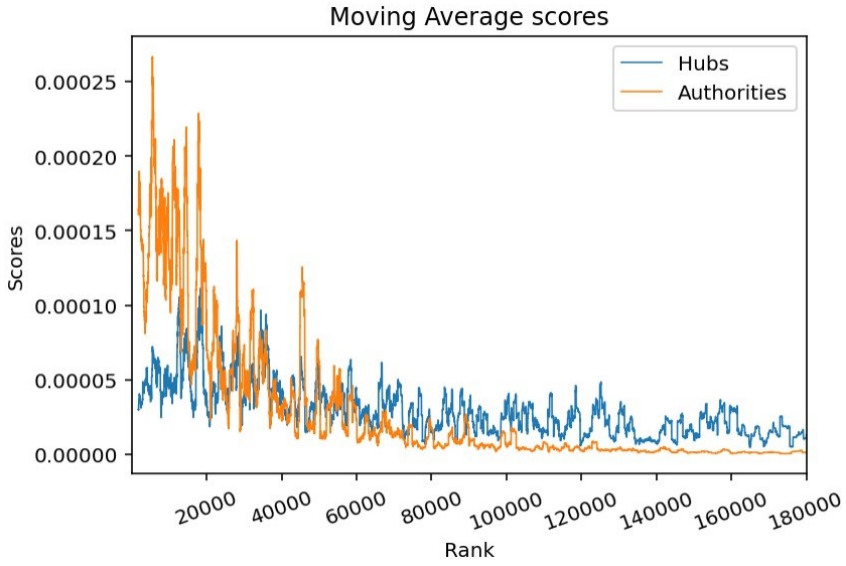

Figure 12: Sale rank compared with both authorities and hubs. We performed a moving average across the node scores. In the first case we see that the most sold product are related to the highest authorities scores, while in the latter this correlation seems to be less relevant.

3.5 Centrality measures

In a network, the centrality relates to the most important or central vertices within a graph. There exist a number of different measures, one is the HITS approach that has been already discussed in section 3.4. Another important centrality measure is based on the PageRank, which is the algorithm used by Google Search to rank the web pages results of the search engine. The PageRank determines the importance of the website by counting the number and quality of links to the page. The main algorithm equation, which express the probability one might jump to a random page, is given by the following expression:

$$\boldsymbol{p_{t+1}} = c\boldsymbol{M}\boldsymbol{P_t} + (1-c)\boldsymbol{q}$$

where $\boldsymbol{p_t}$ is the stochastic PageRank vector, $\boldsymbol{M}$ is the normalized adjacency matrix, c is the damping factor, which is usually equal to 0.85, meaning that 85% of the times we move to one of the links of the page and $1-c$ is the remaining 15% of times one moves to a random page according the probability vector $\boldsymbol{q}$ independent of the node you currently are in. In the networks analysis the web pages are represented by the nodes and the PageRank centrality return the ranking of the nodes in the graph based on the structure of incoming links.

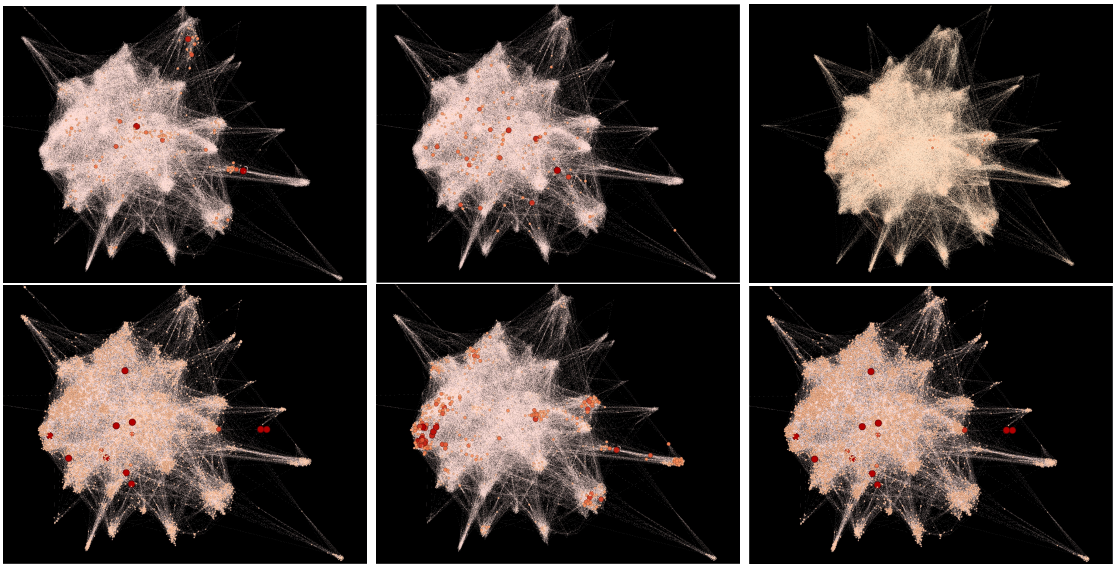

Figure 13: (top-left) PageRank centrality, (top-middle) betweeness centrality, (top-right) degree centrality, (bottom-left) closeness centrality, (bottom-middle) eigenvector centrality, (bottom-right) harmonic centrality.

Considering instead a more general class of metrics, we can define the following centrality measures:

- *Degree centrality*: it is simply the measure of the degree of a node, so the number of neighbors it has. Given the adjacency matrix A of the network, the formula for a node i is: $D_i = \sum_{j=i}^{N} A_{ij}$
- *Closeness centrality*: it is based on the idea that the node which are easiest to reach are the best one for spreading information. It is indeed calculated as the reciprocal

of the sum of the length of the shortest path between the node and all the other nodes in the graph: $C(x) = \frac{1}{\sum_y d(x,y)}$ with $d(x,y)$ distance between vertices x and y.

- *Betweenness centrality*: it is a measure that gives importance to nodes that act as bridges between groups of other vertices, for a node i it is defined by the relation $g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(i)}{\sigma_{st}}$ with σ_{st} the total number of shortest paths from note s to node t and $\sigma_{st}(i)$ the number of those paths that pass though the node i.

- *Eigenvector centrality*: it is a measure in which a vertex is viewed as important if it is linked to other which are also important. In particular the eingenvector centrality of a vertex i is proportional to the sum of the eigenvector centralities of the vertex it is connected to: $e_i = \frac{1}{\rho} \sum_{j=1}^{N} A_{ij} e_j$ with A_{ij} element of the adjacency matrix and ρ a constant.

- *Harmonic centrality*: it is an estimate of the sum of the reciprocal distances $H(x) = \sum_{y \neq x} \frac{1}{d(x,y)}$. Mostly used in the case of weakly connected network to replace the centrality measure that could be zero.

The graphs corresponding to these measures are showed in 13, they have been obtain using Gephi. The computation of the centrality measures with the function implemented in NetworkX requires in fact a too long computational time, however these parameters are of use if we want to study more in-depth our network. This is the reason we've decided to create a filtered graph based on the PageRank, which is usually the benchmark for this type of operation . In Figure 14 we can observe the PageRank values for each of the node of the original network, we selected all those nodes that have a PageRank values above of 9.43^{-5} , which is 5% of the total network: as a results the filtered network has 1664 nodes and 16677 edges.

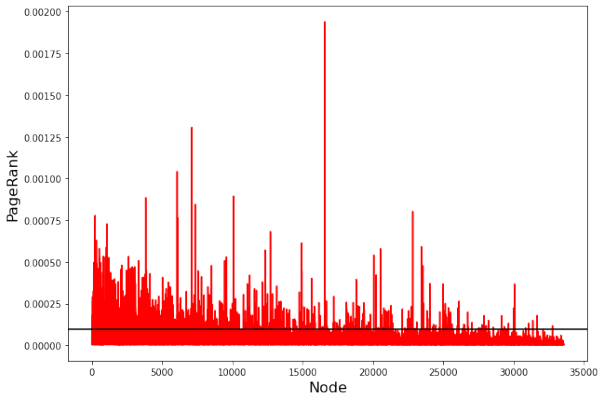

Figure 14: PageRank of each node of the original network with the set threshold (black line).

With a reasonable computational time, we were indeed able to calculate all the different centrality measures using NetworkX, the graphs are reported in the Figure 15.

3.6 Robustness

The robustness of a network can be define as its ability to maintain the network structure under failures and attack. Considering the Amazon Co-Purchasing network this could be

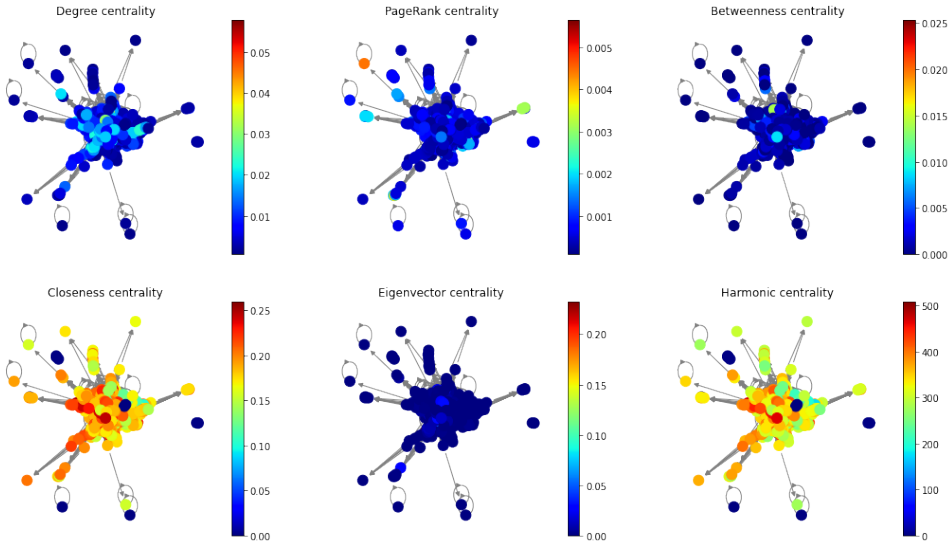

Figure 15: Centrality measures graph representation for the filtered network.

interpreted as the effect that the temporary unavailability of a product, or the removal of a product from the market, could cause on sales.

The failures of the network can be modeled by a random removal of the nodes while an attack it is usually constituted by a targeted removal of specific nodes. This approach find a reasoning in percolation theory which describes mathematically the problem: we can study in fact the dependence of the probability that a node is part of the giant component P_∞ (the connected components of the graph that contains a significant fraction of the entire nodes in the network) with respect to f, the fraction of nodes removed from the network. What we observe is a non gradual fragmentation process of the network characterized by a critical threshold f_c after which P_∞ drops to zero and the network breaks into disconnected component. Moreover from the Molloy-reed criterion, which states that a randomly wired network has giant components if the in-homogeneity ratio $\boldsymbol{k} > 2$ with $\boldsymbol{k} = \langle k^2 \rangle / \langle k \rangle$, one can find the following expression for the breaking point $f_c = 1 - \frac{1}{\boldsymbol{k}-1}$ [6]. For the filtered Amazon Co-Purchasing network, $\boldsymbol{k}$ is bigger than two, so we do have giants components, while the theoretical breakdown point is $f_c = 0.95$ which means that we need to remove 95% of nodes to fragment the network.

In Figure 16 we can observe the trend of the the giant components as a function of the removed nodes, we have considered the following types of attack: random removal of nodes, attack based on the sorted values of degree centrality, betweenness centrality and PageRank centrality. The network considered is the filtered one, since the centrality measures are needed. To estimate the critical fraction of nodes removed f_c we have instead considered both the point where the giant components first drops below 5% and 1%, the results are reported in table 5.

Immediately, one can notice the difference in the random failures trend with respect to the attack based on the centrality measures. Having an insight of the structure of the network, as in the latter case, means we are removing significant nodes; the robustness of the network is weaker as proved by the betweenness curve which has the faster descend and drops significantly with only 40% of removed nodes. Notice that in the 1% drop case the PageRank curve reaches before the critical value and so goes faster to zero, even with

Type of attack	f_c - 5%	f_c - 1%
Random	0.87	0.92
Betweeness	0.61	0.83
Degree	0.79	0.86
PageRank	0.67	0.76

Table 5: Critical fraction of nodes removed f_c for the two drop percentages for the different types of attacks

respect to the betweenness curve which, we could assert, has a slowdown in the end.

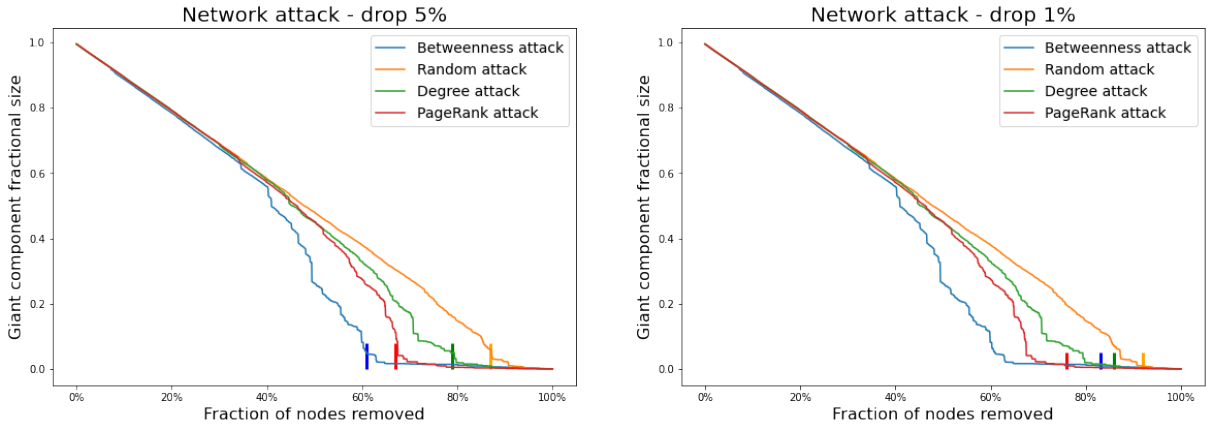

Figure 16: Giant component fractional size as a function of the fraction of removed nodes for different types of attack and for two different drop rate, 5% on the left and 1% on the right.

3.7 Community Detection

Community structure is an important feature of complex networks, which indicates the fact that nodes are gathered into several groups. Each group is a community of nodes where the density of edges within communities is higher than among communities. Indeed, community detection, also called graph partition, helps us to reveal the hidden relations among the nodes in the network. Several techniques have been utilized to investigate community structures of networks during last years, mainly based on spectral clustering techniques or on network modularity optimizations techniques. In this report we focus on the latter and we implement some algorithms of community detection, such as *greedy modularity optimization* and *Girvan–Newman algorithm*.

- **Louvain method - greedy modularity maximization**

  A classic way of detecting communities is to find a partition of the vertex set that maximizes an optimization function. One of the most famous optimization function is called *modularity*. This function provides a way to value the existence of an edge between two vertices of an undirected network by comparing it with the probability of having such an edge in a random model following the same degree distribution than the original network. For instance, an edge between two vertices of large degree is not surprising, and thus does not contribute much to the modularity of a given

partition, whereas an edge between two vertices of small degree is more surprising. Formally, the modularity Q of a partition C of an undirected graph $G = (V, E)$ is defined as follows :

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{1}$$

where m stands for the number of edges of G, A_{ij} represents the weight of the edge between i and j (set to 0 if such an edge does not exist), k_i is the degree of vertex i (i.e. the number of neighbors of i), c_i is the community to which vertex i belongs and the δ-function $\delta(u, v)$ is defined as 1 if $u = v$, and 0 otherwise.

This formula [eq.(1)] could be extended to the directed case motivated by the the fact that if two vertices u and v have small in-degree (large out-degree) and small out-degree (large in-degree), then having an *arc* from v to u should be considered more surprising than having an *arc* from u to v. Taking this into account, the previous definition for directed modularity of a partition of a directed network can be easily update substituting $k_i \to k_i^{in}$ and $k_j \to k_j^{out}$ [7].

We focus mainly on the Louvain's algorithm that consist in maximizing the modularity. The algorithm is the same for both the classic and directed versions of modularity. In practice, it relies on a greedy procedure: starting from any partition of the vertices (usually the partition into singletons), the algorithm tries to increase the value of modularity by moving vertices from their community to any other neighbor one. In other words, the algorithm computes the gain of modularity obtained by adding vertex i to community C as follows (for the undirected case):

$$\Delta Q = \left[\frac{\sum_{in} + k_i^C}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right) \right] \tag{2}$$

where k_i^C denotes the degree of node i in community C, $\sum_{in}$ in the number of edges contained in community C and $\sum_{tot}$ the total number of edges incident to community C.
The directed version is exactly the same as the previous formula in which $\sum_{tot}$ splits in $\sum_{tot}^{in}$ and $\sum_{tot}^{out}$ that denotes the number of in-going and out-going arcs incident to community C.

The algorithm does a similar calculation to compute the gain obtained by removing vertex i from its own community C_i in a first place. The algorithm carries on as long as it exists a move that improves the value of modularity.

In Figure 17 is represented the outcome of applying greedy-modularity maximization on the original graph G. The output of the algorithm found something like 389 communities that are represented in the plot by a different color.

Since it's difficult to find out some meaningful structures and also the computational time becomes not irrelevant, we decided to filtered the graph based on PageRank, as previously mentioned in section 2.5. In practice we had to manage with a tiny subgraph composed by 1664 nodes, which turned out to be a good choice. Indeed, what we obtain is represented in the plot in Figure 18 that looks quite a bit less confusing.

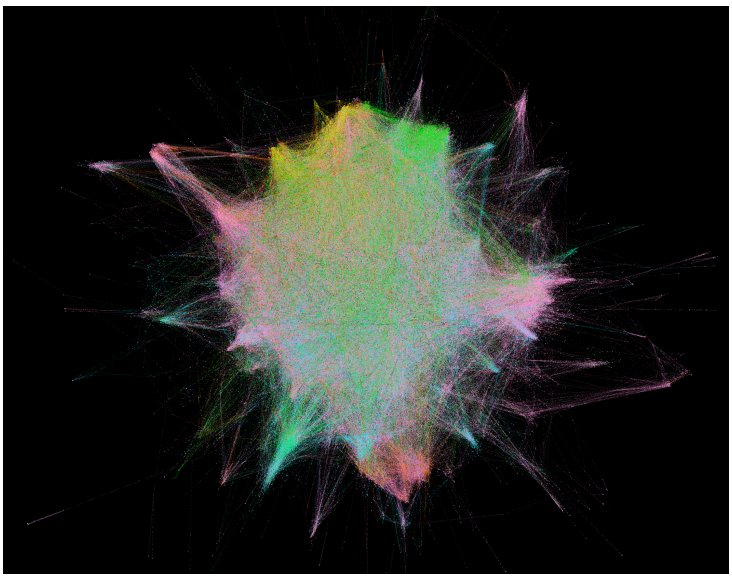

Figure 17: Entire network in which each color correspond to a community find by the Louvain method. In total there has been found 389 communities

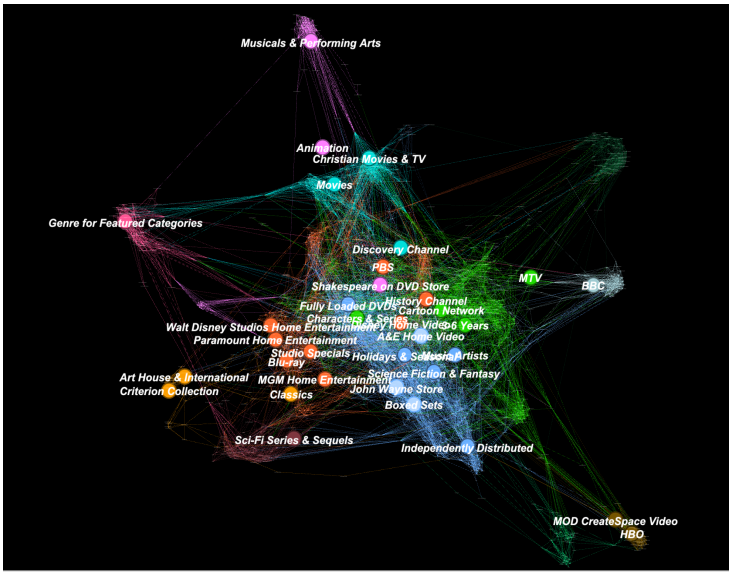

Figure 18: Reduced network composed by 1664 nodes, the colors represent the community found by the algorithm while the big colored dots are the "supernodes" associated to a given main category. Their colors are based on the specific community found by the Louvain method.

This time applying again the same greedy modularity optimization algorithm we end up with 18 communities.

Looking deeper we try to investigate if there is a sort of correspondence with our data structure. In this way, we have search a match between the categories and the founded communities. We decided to build a sort of *"supernodes"* structure in which we condense multiple nodes into a single "bigger" one. Each of these "supernodes" has been associated to a given category (i.e. the first macro category of each entries). Practically, we filtered the dataset for each macro category and we've assigned the

role of "supernode" to the one with the highest pagerank value. Computing this on the entire dataset, gives us a value of 66 *(macro) categories*. After reducing the size of the network about of the 95%, we find out that more than half of the supernodes remains (i.e. 34 supernodes) and this sound quite impressive. Returning to Figure 18 it could be seen how each big dot represent an assigned supernodes while the colors of the network represent the different communities retrieved by the algorithm. We could notice that some of the "supernodes" appear correctly in correspondence of a detected community, meaning that the communities found by this procedure has a good correspondence with the main category of the dataset. On the other hand, multiple "supernodes" share the same colour meaning that probably the there are different type of structures behind the communities. Given a look closer to the network by zooming into it, in Figure 19 there are some example of how the categories ("supernodes") have a positive match with the communities. Furthermore, we find out that sometimes the output communities of the algorithm doesn't not correspond to a proper main category (i.e. not correspondence with the "supernodes") but instead to a *specific type of movies*, like in Figure 20 for the *musical movies* and for the *"Doctor Who" series*.

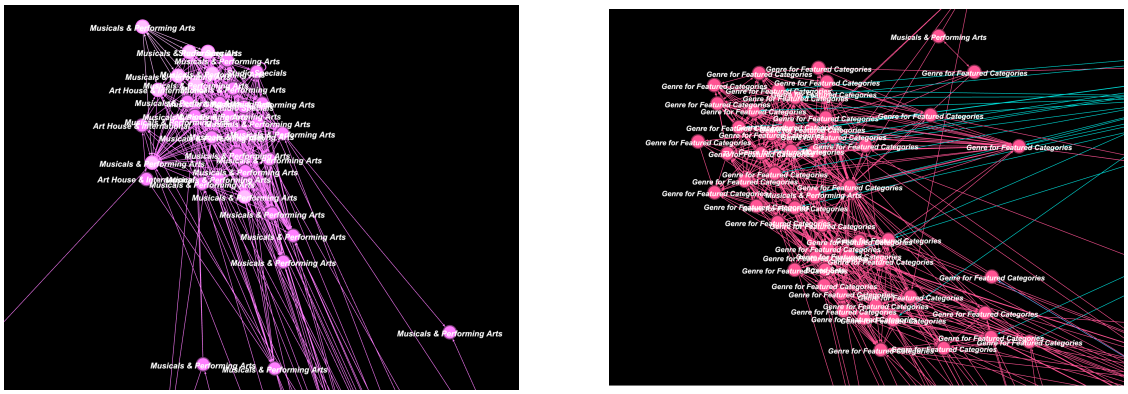

Figure 19: Example of positive matches between community and main category: (*left*) "Musical and performing art" category; (*right*) "Genre for Feature Categories" category.

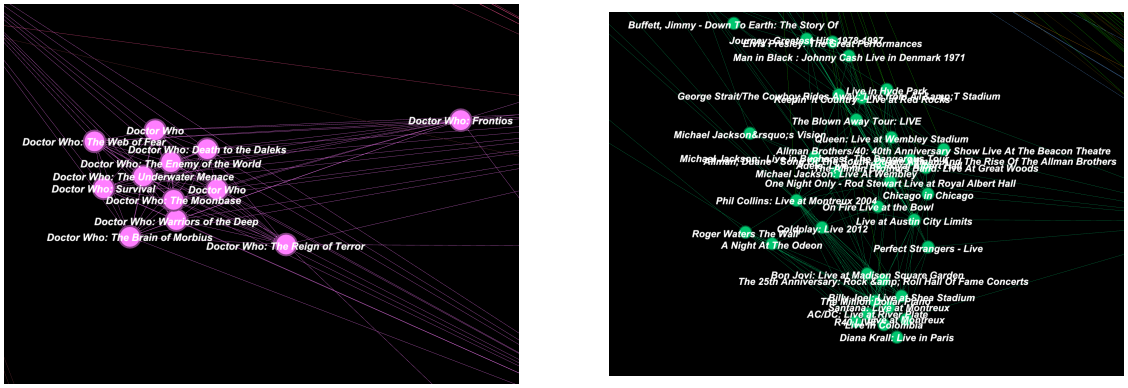

Figure 20: Example of positive matches between community and specific type of movies such as the science fiction "Doctor Who" (*left*) and the musical movies of live concerts (*right*).

- **Girvan–Newman algorithm**

  This well-known algorithm relies on the iterative elimination of edges that have the highest number of shortest paths between nodes passing through them. By removing edges from the graph one-by-one, the network breaks down into smaller pieces (communities).

  The Girvan Newman strategy [8] is based on betweenness centrality as a measure of significance and on modularity Q as evaluation of goodness.

  The following steps summarize the method:

  1 - Computing betweenness for all edges in the network.

  2 - Remove the edge with the highest betweenness.

  3 - Re-computing betweenness for all affected edges by removal.

  4 - Repeat the steps *2, 3* until no edges remain within the network.

  The definition of the partition of "goodness" is obtained by searching the maximum of modularity Q, given in [eq.(1)].

  This algorithm, unfortunately, require a very long computational time due to his high computational complexity. This make it inappropriate for a large network and it translates to an infeasible procedure on applying this to the entire network, caused mainly by our lack of computational resources. As a consequence, we restrict its application only to the filtered network (1664 nodes) as we've done previously.

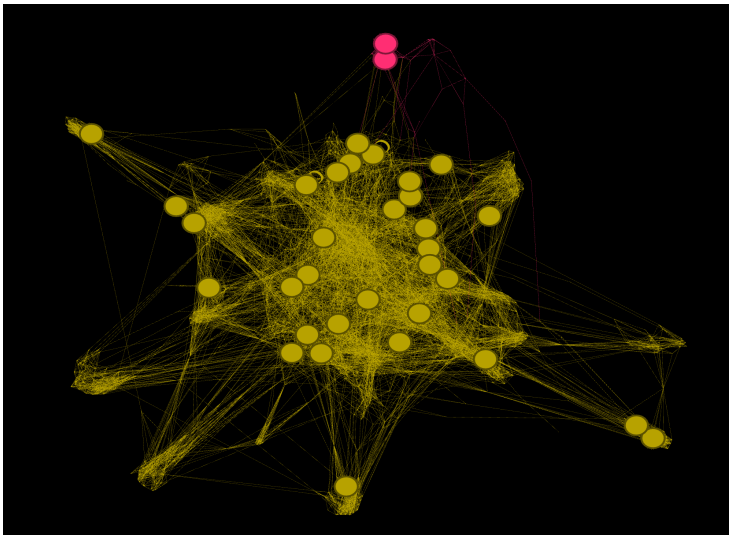

Figure 21: Application of the Girvan-Newman algorithm to the reduce network. It found 10 communities (one for each color) but the vast majority of the nodes belong to only one of them. The big colored dots instead are the "supernodes" associated to a given main category. Their color are based on the specific community found by the algorithm

However, this method has some resolution limits, means that it tends to detect super communities, subsequently it considerably slow down the algorithm. Indeed,

the output of the algorithm gives us 6 different communities but one of them consist in the vast majority of the entire network. This could be seen better in Figure 21 where we could notice that most part of the network belong to only one "super" community (each color is one community) while the others are very small and quite irrelevant.

Given these results, we decided to consider the first *greedy modularity optimization* algorithm as the best method for this particular task, since it finds out a more coherent partition of the communities. Furthermore we found how these communities could be interpreted, looking at some of the attributes of our data (i.e. the "category" and the "title" columns of the dataset)

3.8 Link Prediction

The idea behind the link prediction implementation was to find something interesting that was also business related. Of course the simple *'costumer item suggestion'* falls under this category, but here we want to exploit the network properties to have a much better final result. The goal is to predict what to suggest to a potential customer given the fact that he/she is viewing a certain item (a certain node of the graph). Usually this is done with deep neural network by learning the customer 'habits', but here instead we seek to find other potentially useful nodes in the graph as suggestion. The core problem was to generate some random edges that do not exist and label them as 0, while retaining the true edges and label them as 1. Then training a simple machine learning classifier and use it to suggest other items. It should be noticed that, to further improve accuracy, we could also use deep NN.

3.8.1 Algorithms

- **Jaccard coefficent**

  The Jaccard coefficient gives a value that represents the similarity of the neighborhoods of two vertices. Given a pair of vertices U and V we represent Jaccard as the intersection of the neighborhoods of U and V, divided by the union of the neighborhoods of the two vertices.

  $$J(x, y) = \frac{x \cap y}{|x \cup y|}$$

  From a computational standpoint most of the work comes from computing the intersection of U and V. This is because to compute the union you can simply take the size of both the neighborhoods, add them together and then subtract by the size of the intersection, so that the shared vertices will only be counted once. This was implemented with the *NetworkX* library, but of course it was to zero in the case of generated fake edges between nodes.

- **Resource allocation**

  The Resource Allocation algorithm was introduced in 2009 by Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang as part of a study to predict links in various networks. It

is computed using the following formula:

$$RA(x,y) = \sum_{u \in N(x) \cap N(y)} \frac{1}{|N(u)|}$$

where $N(u)$ is the set of nodes adjacent to u. A value of 0 indicates that two nodes are not close, while higher values indicate nodes are closer. As previously done, this algorithm was implemented with the *NetworkX* library but of course it has been set to zero in the case of generated fake edges between nodes.

- **Preferential attachment** [9]
  Preferential Attachment is a measure used to compute the closeness of nodes, based on their shared neighbors. This means that the more connected a node is, the more likely it is to receive new links. It's computed using the following formula:

$$PA(x,y) = |N(x)| \cdot |N(y)|$$

  where $N(u)$ is the set of nodes adjacent to u. A value of 0 indicates that two nodes are not close, while higher values indicate that nodes are closer. In this case for fake generated edges this coefficient may not be zero, so we calculated it for all the edgelist.

- **Node2Vec** [10]
  Node2vec algorithm uses skip-gram with negative sampling (SGNS, this is related to the common *word2vec* implementation used to encoded words in a latent space). To sum it up, the *node2vec* algorithm uses random walks to generate a number of sentences starting from each node in the graph[11]. The walk length parameter controls the length of the sentence. Once the sentences are generated using random walks, the algorithm inputs them into the SGNS model and retrieve the hidden layer weights as node embeddings.
  After calculating the latent representation we used the **cosine similarity** to have a feeling of the distance between two connected nodes.

$$cos(x,y) = \frac{x \cdot y}{||x|| \cdot ||y||}$$

  In addition to this algorithm a latent representation of the edges using the **Hadamar Embedder**[12] and then reduce the number of dimension for the latent space with PCA to retain $\sim 70\%$ of the variance.

3.8.2 Results

To check if the network properties are useful in order to suggest the correct items to the customer we tried two different analysis: the first using only the item attributes like *rank, price, category and brand* (dividing brands and categories for source and target nodes); the second exploiting the network properties in addition to the nodes attributes. So for the second analysis we also considered the above algorithms (Node2Vec, Jaccard coefficient, Preferential attachment, Resource allocation, Hadamar embedding). It has to be noted that jaccard coefficient and resource allocation are set to zero for fake nodes, this

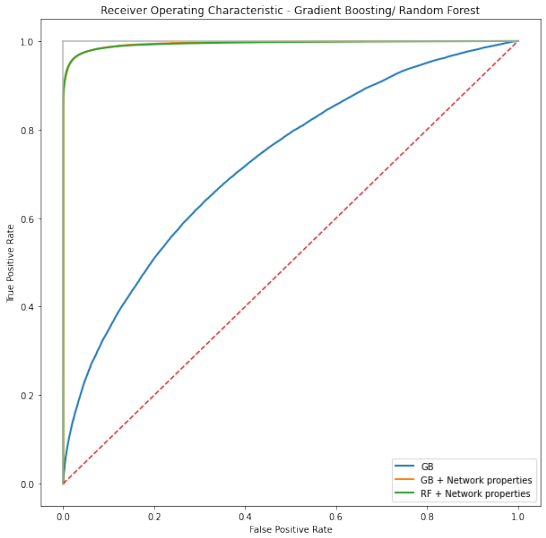

Figure 22: ROC graph. We clearly see that without the network features the ROC score is $\sim$ 30% lower (blue line), while exploiting the graph structure either with random forest and gradient boosting the prediction reaches a 96% ROC score. The red dotted line indicates a random classifier.

may lead to a trivial classifying result. So we also implemented another version without these two attributes but the results where nearly the same.
Since we have a classification problem it is natural to check whether the accuracy of the machine learning algorithms is high. We implemented a simple Gradient Boosting classifier[13]. We also applied a decision tree classifier to also have a sense of the importance of the single attribute in the analysis.
The results are very interesting, in fact we see that without the network properties the accuracy for predicting fake nodes is $\sim$ 67% (not random but in general mostly inaccurate). While exploiting the network properties we see that the accuracy bumps up to $\sim$ 97%, this is astonishing. In the first implementation the decision tree classifier tells us that the main features by importance are the rank with 26% and the price and the brand (with 15% each). So, as we expect people often buy items paying attention to these features, weighting their decision w.r.t. rank and/or brand.
In the latter implementation the main features in order of importance are the cosine similarity, resource allocation ecc., the first dimension of the PCA for the Hadamar edge embedding. So interestingly enough people actually buy stuff based on other co-purchasing habits.

This high accuracy essentially means two possible things: the first (business related) is that exploiting the network properties highly increases the probability of suggesting something that the customer is actually interested in buying (and this basically increases the whole sales). The second is that comparing our first 10 suggested results ordered by the highest cosine similarity we see common elements w.r.t. the "also view" feature attribute (the viewed items the customer clicked on after buying a certain product). Thus, also Amazon itself exploits some kind of graph theory algorithms to suggest items to the customer. This enforces the concept that using the graph structure is a very powerful tool for this kind of business related task. Of course, it is important to say that in general

Metrics	Node attr.	Newtork prop.
Accuracy	67	97
ROC score	66	97

Table 6: Metrics table where we compare the accuracy and the ROC scores between the usual node attribute implementation and the implementation exploiting also the network properties.This is the result with the gradient boosting, however also with the decision tress the results are the same.

companies like Amazon do not only rely on the graph structure but they add many more layers of complexity to the algorithm through deep learning for example.

% importance	Node attr.	Newtork prop.
Brand	15	≤ 5
Price	16	≤ 5
Rank	26	≤ 5
Category	13	≤ 5
Cosine sim.	-	28
Resource all.	-	25
Pref. att.	-	7
Jaccard	-	23
$1^{st} dimEdge$	-	≤ 5

Table 7: Relative importance of the node edges attributes. We clearly see that when exploiting the graph structure these features become much more useful w.r.t. the standard brand, price and rank features.

4 Conclusions

In this work we presented an analysis of the *Amazon Co-purchasing* network. The network was related to the Movies & Tv section of items and has about 30'000 nodes and 400'000 edges. We started with an estimation of the degree and an assortativity analysis, followed by centrality measures and some consideration behind. Due to limited computational resources we filtered the network using PageRank scores to keep only the 5% of the whole data. From here we studied the robustness of the network using betweenness, PageRank and degree centrality, comparing them to a random node removal as we see in Figure 16. We clearly see how attacking the network removing the 'bridge' nodes breaks it down much faster. For what concerns community detection analysis, we used the *Louvain* and *Girvan-Newman* algorithm to cluster the different graph parts, comparing them to the real node category attributes where we interestingly found some correspondence. The last part was done with the entirety of the network and is dedicated to a link prediction task in order to find something that was also business related for this project. Results shows how exploiting the network feature we are able to perform a good 'customer suggestion' job. This is proven comparing our prediction with the *also view* attributes of the nodes. It also proves that Amazon itself uses the whole network properties for the task.

5 Contributions

The whole project has been developed by all the components of the group, but we have respectively focused more on:

Fontana Francesco: Community detection and data preprocessing.
Giorgetti Sabrina: Centrality measures and robustness.
Latora Federica: Degree study and assortativity.
Ziliotto Filippo: Link prediction, authorities & hubs.

References

[1] J. M. J. Ni, J. Li, "Empirical methods in natural language processing (emnlp)," 2019.

[2] A. Barabàsi, "Network science," 2016, ch. 2 "Graph Theory".

[3] ——, "Network science," 2016, ch. 7 "Degree Correlation".

[4] "Node assortativity coefficients and correlation measures," https://networkx.org/nx-guides/content/algorithms/assortativity/correlation.html.

[5] P. Patel and K. Patel, "A review of pagerank and hits algorithms," *International Journal of Advance Research in Engineering, Science Technology*, vol. 2, pp. 2394–2444, 02 2015.

[6] A. Barabàsi, "Robustness in scale-free networks, scientific american," 2003. [Online]. Available: https://barabasi.com/f/619.pdf

[7] A. P. Nicolas Dugué, "Directed louvain: Maximizing modularity in directed networks," *[Research Report] Université d'Orléans*, 2015.

[8] A. Almukhtar and E. Al-Shamery, "Greedy modularity graph clustering for community detection of large co-authorship network," *International Journal of Engineering Technology*, vol. 7, p. 857, 11 2018.

[9] G. G. Piva, F. L. Ribeiro, and A. S. Mata, "Networks with growth and preferential attachment: Modeling and applications," 2020.

[10] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," 2016.

[11] E. Cohen, "node2vec: Embeddings for graph data," 2018. [Online]. Available: https://towardsdatascience.com/node2vec-embeddings-for-graph-data-32a866340fef

[12] D. Berberidis and G. B. Giannakis, "Node embedding with adaptive similarities for scalable learning over graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 2, pp. 637–650, 2021.

[13] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in neurorobotics*, vol. 7, p. 21, 12 2013.