UNIVERSITÀ DI PISA

Final report in

Social Network Analysis

TripAdvisor Tuscany network: restaurants and reviews

Ilenia Bari

i.bari@studenti.unipi.it

Student ID: 590445

Ivan E. Ogando Perez

i.ogandoperez@studenti.unipi.it

Student ID: 606561

Michele Andreucci

m.andreucci4@studenti.unipi.it

Student ID: 628505

Carlo Alberto Carucciu

c.carucciu@studenti.unipi.it

Student ID: 533967

Accademic Year 2020/2021

Trip Advisor Tuscany Network

ABSTRACT

The TripAdvisor analysis for Tuscanian restaurants, allows us to create an undirected and weighted network in which the nodes are represented by the structure, the edges by the sharing of the reviewers and the weight is related to the number of these. Moreover we were able to track chronologically the movements of all Trip Advisor reviewer among the restaurants of the region. So we use the information gathered build a directed multigraph and see how the flows of people moved during the pandemic period of the 2020-2021.[1]

KEYWORDS

Social Network Analysis, Trip Advisor, Tuscany, Florence, Restaurants, Covid, Tourism, Italy, cuisine, Reviews, Time Respecting Path, Diffusion Network

ACM Reference Format:
. 2021. Trip Advisor Tuscany Network. In *Social Network Analysis '21.* ACM, Pisa, PI, ITALY, 19 pages.

1 INTRODUCTION

In this paper will be illustrated the phase to build the network, and the analysis on it. After a first step of crawling we obtain a undirected network. The objective of our project is to analyze the relationship among the restaurants and bars in Tuscany, for which there are reviews on TripAdvisor. Study the characteristics of the net, computing the path, the centrality analysis and the clustering coefficient, comparing with synthetic networks. Discover, also, if there are communities or clusters in the graph and which they are, and study the diffusion model of our network. A part is dedicated to the compute of the link prediction using two different approaches. To conclude our work we spend a part of the report to talk about open questions. In this part we consider the reviews on the platform for the Covid period. A directed multigraph has been built by using the dates of the reviews and the reviewer. External informations and different approaches has been useful to complete out information in making multiple analysis. We analyze the characteristics of this new network, considering Covid measures, the positive case trend and user behaviour.

2 DATA COLLECTION

Our idea was to collect data about some touristic structures, specifically restaurants, and find a social connection between them. We opted to be as thorough as possible in the data collection phase, excluding no restaurants. Given limitations of resources to process the amount of data, we limited our research in a geographical manner, focusing on Tuscany only. This also helped to avoid taking a random sample.

Selected Data Sources

For our purposes there were a multitude of suitable websites. In fact in last years each online tourism service has been trying to integrate more and more social features such comments and reviews to their offerings. After reviewing some of the most famous websites of this type (Airbnb, Tripadvisor, Booking), we noted that Tripadvisor was the most developed for restaurants information, and presented facilities at the moment of scraping the information, unlike others of the options (to scrape airbnb for example, it was necessary to use Selenium with a web driver). The strategy was to visit the pages of restaurants in Tuscany, province by province, since it was possible to have for a specific "locality" in Tripadvisor, the full list all the restaurants (plus coffees, etc...) of this location. Once collected, the list of restaurants with respective urls, the second phase consisted in visit each individual web-page and collect all the reviews for each restaurant. The reviews were used in order to build a network where nodes are restaurants and edges are the reviews.[2]

[1] **Project Repositories**
Data Collection:
https://github.com/sna-unipi/2021---final-project-andreucci_bari_carrucciu_ogando/tree/main/data_collection
Analytical Tasks:
https://github.com/sna-unipi/2021---final-project-andreucci_bari_carrucciu_ogando
Report:
https://github.com/sna-unipi/2021---final-project-andreucci_bari_carrucciu_ogando

[2] Two restaurants are connected if the same user reviewed both.

SNA '21, 2020/21, University of Pisa, Italy

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM. . . $0.00

Crawling Methodology and Assumptions

Different approaches and methodologies were considered, comparing the cost and benefit of each. At the end, we opted to build a crawler with Scrapy. After research and trials, it was deemed as the fastest tool for crawling using python. We were able to collect satisfactory data without necessity of Selenium, the option that offered the largest array of possibilities but demanded a excessive amount of time and resources. So three scrapy spyders were built:

- The first one iterated over the geographical page-list of restaurants, and we ran it once for each province, saving the data in csv format. Information such as name, province, etc...
- The second iterated over the links of the csv's, visiting the main page of each restaurant and gathering more information about it, enriching the ones we already have. We are so able to correct the information previous collected, adding also details such as cuisine types, special diets offered, and some boolean features. Furthermore we retrieve detailed geographical information: latitude and longitude.
- The third one was for reviews, for a given restaurant collects all reviews and saves those in a json format. We iterate over restaurants by another piece of code, running the spider in a loop and writing one json file for each restaurant, subdivided in folders. In particular keeping track of username (when correct, otherwise a combination of information is used as id key) and date.

The process was easily parallelizable, given that it was naturally divided by province. Furthermore records where transcripted one by one; so in case of an error or an unexpected stop, we were able to restart from the interruption. In addition to this, considering the time necessary to complete the data collection, we leaned on the free machines offered by amazon web service. Taking advantage of their processing power and of their web connection. Several days were necessary to complete this operation.

At the end, after cleaning and reorganising the data, the dataset contained 15864 restaurants(having at least one connection) with a total of 2 millions reviews.

3 NETWORK CHARACTERIZATION

Network building

At this point we were theoretically able to build a bipartite network and project it into a unipartite network using an appropriate technique. However, given some technical issues such as limited memory and time to process the size it would have had. The most suitable solution was to project the network from the data structures we already had: iterating over the json files we are able to build a weighted network. Different methodologies are proposed by related literature ([12], [1]) to choose which weights to use. We took in consideration both simple weight (number of mutual reviewers) and Jaccard's coefficient (|mutual| / |union|). The reason being that the first one is an absolute measure, that gives importance to most reviewed locals. The second one is more reliable in terms of similarity between two restaurants. The projected network had in total 6.955.000 edges, resulting infeasible to be used for our analysis. The best way, a posteriori, to lighten the graph, was to use backboning as suggested in paper [2], an approach that removes only superfluous edges. This way allowed us to sample the network by pruning edges, without disconnecting the network itself. With this approach, we opted to contain the size of our network to 136.000 edges.

Synthetic Networks

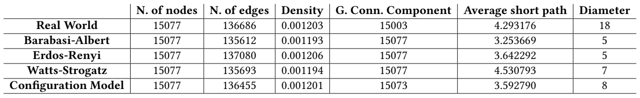
For comparing our network, we built 4 different types of synthetic networks:

- Erdos-Renyi **(ER)**, took as paramether the number of nodes and the density;
- Barabasi-Albert **(BA)**, where we gave as paramether the number of nodes, and the average degree of the real network divided by 2;
- Watts-Strogats **(WS)**, took the number of nodes and their average degree, and as probability took the standard deviation of the nodes degree normalize by the greatest degree of a node;
- Configuration Model **(CM)**, that took the identical degree sequence extracted by the real network.

Networks were built using Networkx library, where we were able to define parameters basing on the real one network.

Dimensions and density

All the networks considered had 15077 nodes, and a similar number of links. Nodes in ER, BA and WS, presented the complete giant component, while in the real world network we had a giant component of 15003 nodes, 6 component of 3 nodes each one and 30 components of 2 nodes each one. In the Configuration model network, there were 3 components composed by 15073, 2,2 nodes..



	N. of nodes	N. of edges	Density	G. Conn. Component	Average short path	Diameter
Real World	15077	136686	0.001203	15003	4.293176	18
Barabasi-Albert	15077	135612	0.001193	15077	3.253669	5
Erdos-Renyi	15077	137080	0.001206	15077	3.642292	5
Watts-Strogatz	15077	135693	0.001194	15077	4.530793	7
Configuration Model	15077	136455	0.001201	15073	3.592790	8

Table 1: Networks Statistics

Given the high computational cost necessary to compute all the paths, we opted for using the average shortest path. Which, with the diameter, gave us important information about the structure of the network and confirmed the small world theory. All the average shortest path values are quite low, from 3.25 to 4.53; for the synthetic networks the diameter was represented by a low value for the dimensionality of the network. Only the real world network had a diameter of 18 confirming the small world property. Regarding the connection, the graph appeared to be sparse with density values not higher than 0,001206 (ER Model)

Degree distribution

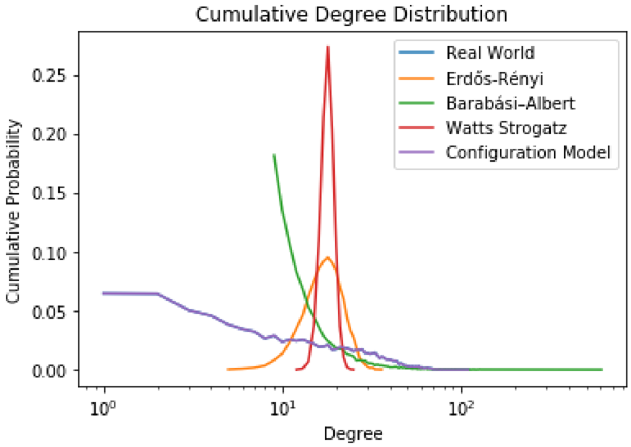


Figure 1: Degree distribution

Observing the degree distribution plot in figure 1, we could tell with ease that WS and ER had a Poisson distribution, while Real World and BA have power law. Considering that BA is an algorithm created to generate synthetic networks scale free using the preferential attachment,it was clearly understood why it had this particular distribution.

	Max	Min	Average	Variance
Real World	110	1	18.1317	235.8660
Barabasi-Albert	610	9	17.9892	494.0137
Watts-Strogatz	25	12	18	2.3174
Erods-Renyi	36	5	18.1840	18.0668
Configuration Model	109	1	18.1011	234.5261

Table 2: Nodes Degree Statistics

Analyzing in detail the values related to the degree of the different networks (Table 2) we can assume that:

- Each average degree was around 18. This was understood given the fact that each net has the same or almost, number of nodes and degrees.
- Configuration model and real world have exactly identical values (the configuration given was based on replicating nodes degree);
- BA, given its characteristics has as max degree a number higher than the other networks.
- In the Real World and BA, looking at the average degree and the variance, there were situations in which some few nodes appeared with an high degree, while many others presented with a low value. This also appeared graphically, where in each considered network, there were peaks that descended rapidly.

Clustering coefficient

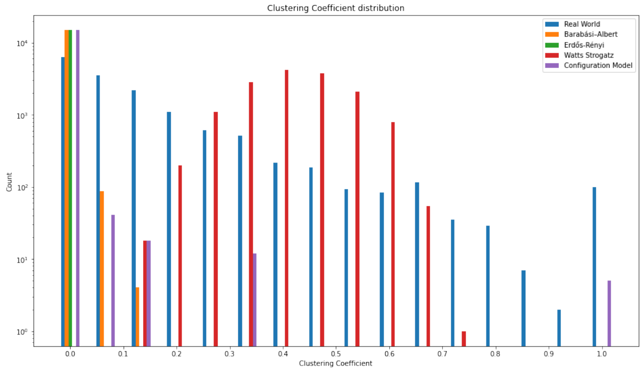


Figure 2: Clustering coefficient distribution

	Max	Min	Average	Variance
Real World	1.0	0.0	0.1277	0.0248
Barabasi-Albert	0.1667	0.0	0.0073	0.00012
Watts-Strogatz	0.7333	0.1522	0,4538	0.0078
Erods-Renyi	0.0444	0.0	0.0012	9.3856
Configuration Model	1.0	0.0	0.0033	0.0004

Table 3: clustering Coefficient Statistics

ER and BA displayed an exceptionally low clustering coefficient. In addition, after looking at the clustering coefficient distribution plot, the majority of the values were concentrated in the first part of the horizontal axis, between the numbers 0 and 0.20. As for the Real World we saw the plus value of the clustering coefficient was 1, this meant that there was at least one triplet that was perfectly related to each other. Therefore, according to the objective of our analysis, there were at least three restaurants / bars that had received reviews from the same people, and others given the minimum value of 0, which had a different type of clientele since none of the vertices are connected to each other. However, it can be seen both from the average value, closer to 0 than to 1, but also from the plot, that most of the clustering coefficients are less than 0.4, few nodes if not none up to 0.8 and then present just over 100 nodes with a clustering coefficient equal to 1. Another issue for the WS whose coefficients are concentrated in the range of values 0.1-0.7. There was no triangle, but no node is totally disconnected from its neighbors, information that we also receive from the minimum coefficient of 0.15.

4 CENTRALITY

To understand the influence of each node in the network, we computed different kinds of centrality measures.

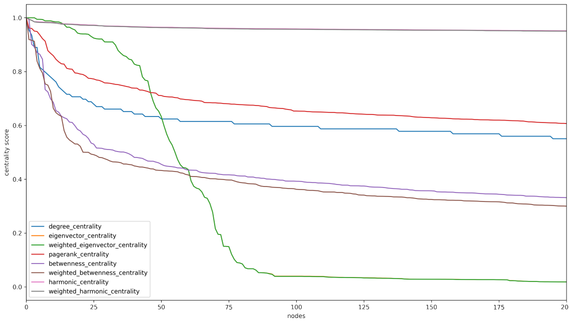
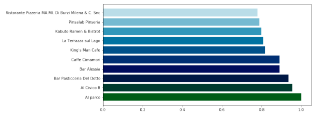
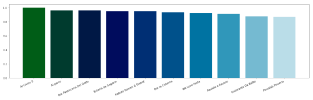


Figure 3: Centrality distribution

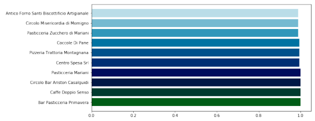
The plot shown (Figure 3) represents the centrality distribution. In this figure we represented only 200 nodes, considered the most representative to our analysis. It can be seen that there were a low number of nodes with high values, less than 25. Around this number the distribution tended to stabilize. Two exceptions were represented by Eigenvector and weighted Eigenvector, when the curve went down until 100 nodes, and then stopped on the descent; and the harmonic centrality, weighted and not, that had a minimum descent and stabilize itself.
To visualize in detail the results of the different algorithms, these are shown under in form of rankings of the top 10 nodes.



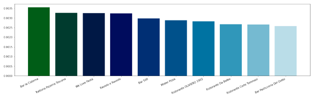
(a) Degree



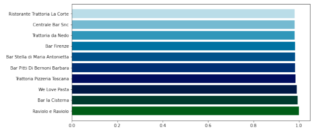
(b) PageRank



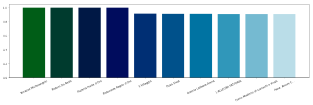
(c) EigenVector



(d) Betweenness



(e) Harmonic



(f) Multicore

The **degree centrality** computation found among the most important nodes, those that have a greater degree. The plot shows first 10 nodes with the highest centrality value. For brevity, we will analyze only the first 5 names: Al Parco (Arezzo) with degree score as 110, Al Circolo 8, degree of 105, Bar Pasticceria del dotto with 103, Bar Alessia e Caffè Cimamori with degree di 98.
Concerning the connectivity-based centralities, they were computed by utilizing the algorithms of PageRank, EigenVector, and Katz centralities. The **PageRank** centrality was also computed on the weighted network and it appeared to confirm, though with lower values, the importance of Al Civico 8, Al Parco, Bar Pasticceria del dotto in the first 3 places, and Kabuto Ramen e Bistrot, Ristorante de Balbo e Pinsalab Pinseria, We Love Pasta, in the rest rank. **EigenVector** centrality identified the first 5 nodes as Bar Pasticceria Primavera, Caffè doppio senso, Circolo Bar Ariston Casalguidi, Pasticceria Mariani e Centro Spesa Srl. There are vertices, which names, never appeared in the first 10 centrality values, while, as we saw analyzing the rest of the algorithms, there are few names that appeared with certain frequency.
As Geometric based centrality estimators, betweenness and harmonic centrality ahve been computed. The **Betweenness** highlighted a greater influence in the path of Bar la Cisterna, Trattoria Pizzeria Toscana, We Love Pasta, Raviolo e Raviolo. For the top 10 positions we found nodes already present in the result of the other algorithms. The **harmonic** centrality confirmed the higher centrality for Raviolo e Raviolo, Bar la Cisterna, We Love Pasta, Trattoria Pizzeria Toscana. On the same level of centrality we found a recurred name in centralities ranking, Ristorante da Balbo. The **multicore** centralities provided by the proposed algorithm in section 7, guesses one or a group of central nodes for each community. So we can see here accordingly with the community detection detailed in section 5 that we have four core nodes7 in the net, distant from each other: Terrazze Michelangelo, Trattoria da Nello, Pizzeria Ponte d’Oro, and Ristorante Ragno d’Oro.
We could conclude that there are a small number of nodes that are very influent in the network and, according with the results of the different algorithms, these nodes are: Al Civico 8, Al Parco, Bar Pasticceria del dotto, Kabuto Ramen & Bistrot, Ristorante de Balbo, Pinsalab Pinseria, We Love Pasta, Ristorante Pizzeria Toscana.

5 COMMUNITY DISCOVERY

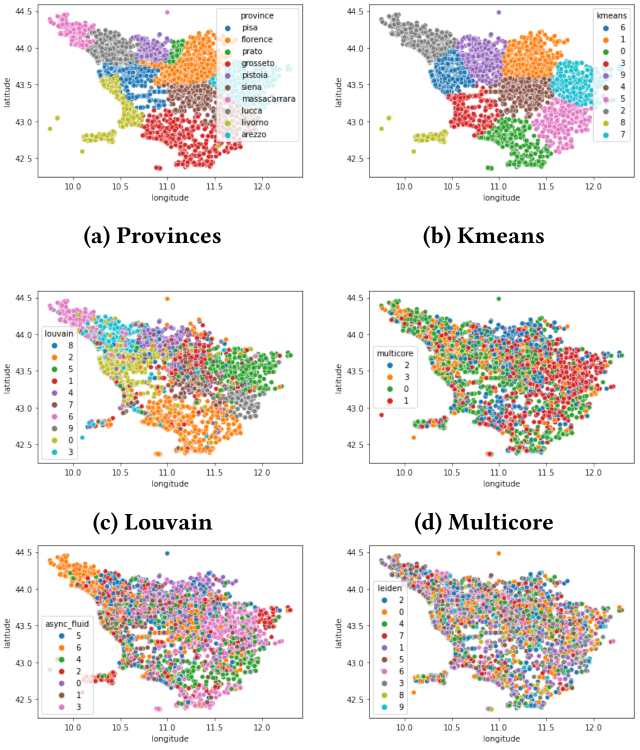
Overview

In this section different known approaches for community discovery are applied on our network. The goal was to find groups inside the network, basing them mostly on their *topology*. In this section we will also display the results obtained with **kmeans** on the geographical coordinates of the nodes.

A new technique described in section 7 named **multicore** has been experimented with and compared with the benchmarks.

	No communities	avg community size	nf1	mutual information
async_fluid_7	7	2143.2857	0.1387	0.217
demon	220	307.8773	0.0005247	NaN
kmeans	10	1500.3	0.4256	0.711
louvain	21	714.43	0.1197	0.428
multicore	4	3750.75	0.0285	0.09
provinces	10	1500.3	1.0	1.0

Table 4: Main Community Algorithms



(a) Provinces

(b) Kmeans

(c) Louvain

(d) Multicore

(e) Async Fluid

(f) Leiden

Figure 5: Communities Geographical Distribution

Moreover we took into consideration the provinces; in fact, the nodes were already labeled from this point of view, and we could create a crisp partition using this attribute. So every goodness measure provided by CDlib, was able to evaluate the crisp partitions obtained by the provinces, and compare the other partitions obtained by the algorithm with this one.

Crisp Partitions

The focus has been on the kind of algorithms, able to label each node, assigning it to a single cluster. So louvain , asyinc fluid, leiden (weighted, unweighted, with starting classes), rbPots and other methods have been applied for this objective. Some of them bared interesting results, other instead were not able to really split the network (greedy, rbpots) , or to guess some consistent class (a lot of small noise communities were found). As part of the crisp partition algorithms, the new algorithm multicore (7) was tested too; by tuning its parameters is possible to obtain interesting results. We were also able to get centralities (see section 4). We were able to show communities found observing the distribution over the geography of Tuscany. We also applied kmeans to the nodes using longitude and latitude; moreover a partition has been created using the province labels of the nodes; both are considerable as crisp partitions too. It is evident in figure 5 how Provinces and kmeans appear pretty similar , with the exception that kmeans isolates *Elba Island*. Louvain communities at the end appear a bit different but geographically well separated, while the multicore is more confusing on a geographic way but still consistent. The same was for asyncfluyd, while we could see rbpots appeared totally random from this point of view.

Overlapping Communities

Angel and Demon algorithms have also been tried on the network. However angel was not able to find a meaningful result so we took in consideration only the demon. It was able to cover a 60% of the nodes with some great communities. For demon, the parameters had a large influence on the results. To decide how to establish them a gird search was applied and the model with the best modularity density was selected (in section 5 we reference that "modularity density" is its Achille heel).

We also built an overlapping partition starting from information about culinary types offered by each single restaurant; 101 communities were detected, one for each cuisine, covering the 90% of the restaurants. However evaluation metrics of section 5 seemed really low and also any mutual information with the demon algorithm, normalized using LFK (paper [6]) and MGH (paper [7]) were almost nonexistent.

Evaluations of communities: Fitness Measures

In this report we include solely the results obtained from the best performing, most interesting and significant solution adopted. All fitness metrics available on *cdlib* were measured in order to select which algorithms were the most useful. These are reported on table 5.

From these measurement wen could note that **kmeans and provinces exhibited, also in this case the same behaviour**. Presenting generally good results and the distinctive characteristic of an high *hub dominance*. **Muticore** had a lower efficiency than async and louvain by a fine margin. We could observe that, in respect to others, there was a **higher degree of randomness** since surprise and significance were

	provinces	kmeans	louvain	multicore	async_fluid_7	demon
AVG EMBEDDEDNESS	0.601829	0.589750	0.699009	0.571367	0.628873	3.874513e-01
AVERAGE INTERNAL DEGREE	10.778917	10.462893	13.097139	9.523394	11.101579	1.288944e+01
AVG TRANSITIVITY	0.168455	0.170878	0.320314	0.162537	0.203799	6.034796e-01
CONDUCTANCE	0.425130	0.442512	0.327495	0.479562	0.387894	6.219615e-01
CUT RATIO	0.000580	0.000601	0.000440	0.000782	0.000522	1.407540e-03
EDGES INSIDE	8012.700000	7980.800000	4168.809524	17808.250000	12277.571429	2.231627e+03
EXPANSION	7.836483	8.149616	6.261093	8.787621	6.698830	2.067839e+01
FRACTION OVER MEDIAN DEGREE	0.480014	0.482276	0.471339	0.478090	0.477606	4.682500e-01
HUB DOMINANCE	0.046773	0.047249	0.268093	0.019055	0.033560	2.651779e-01
INTERNAL EDGE DENSITY	0.002277	0.002353	0.033053	0.000657	0.001272	2.320763e-02
NORMALIZED CUT	0.467402	0.485119	0.345045	0.600484	0.443213	6.451140e-01
MAX ODF	60.200000	58.900000	34.809524	63.250000	44.428571	6.536818e+01
AVG ODF	7.836483	8.149616	6.261093	8.787621	6.698830	2.067839e+01
FLAKE ODF	0.310689	0.342788	0.203239	0.396910	0.280275	7.030001e-01
SCALED DENSITY	7.502440	7.751837	108.884338	2.163292	4.190337	7.645239e+01
SIGNIFICANCE	408643	402579	496710	295663	421808	3.347546e+06
SIZE	1500.3	1500.3	714.428	3750.75	2143.285	3.078773e+02
SURPRISE	61775.45	58277.06	109373.17	8749.83	65066.07	NaN
TRIANGLE PARTICIPATION RATIO	0.679002	0.680727	0.655541	0.623583	0.666123	1.000000e+00

Table 5: Fitness Measure for Communities

lower than for other algorithms. These were signs that the deletion of outliers should be done more carefully in the future. *However we considered that the fitness measures were not weighted. Multicore is extremely dependent on the weight.* **Async, like multicore, was good managing the size of the communities. However louvain obtained, in general, better performances** , particularly on wrt. *average transitivity*, *scaled density* and *internal edge density* (**N.B.** small noise communities got high measurements). Also **demon** gives good metrics, sometimes higher than other algorithms, but it **can't be totally compared with others since its different nature**.

Evaluations of communities: Modularity

Different types of modularities have been examined. Measurements were comparable from figure 6 in which they were scaled in logaritmic base. Erdos-Renyi modularity and Z-modularity rewarded in a clearly way the demon algorithm, and also do the link modularity. Appropriate for overlapping communities evaluation. However Demon communities got the lowest score in newman-girvan modularity,and even its bar did not appeared between the modularity densities cause its score was negative [3](great part of edges cross the borders of communities). Async Fluid, Louvain and kmeans scored comparable results, while multicore lacked. Provinces also had similar results to the better performing ones. However, we highlight that Louvain presents a slight advantage to this group . In modularity density the advantage of Louvain was the most evident: the discussion is the same as for the case of fitness measures (scaled density and internal density).

Further Evaluations

CDLib provides other measures that return a comparison between two groups of communities (*node clustering object* in the library). This make possible to understand in a mathematical way how similar are two different partitions; high score means probably there is some correspondence between any communities from the two partitions. From the table at 4 we want to observe mainly two stuffs:

[3] Despite the optimization, modularity density score for demon algorithm was still -1244.568821 (louvain was 143

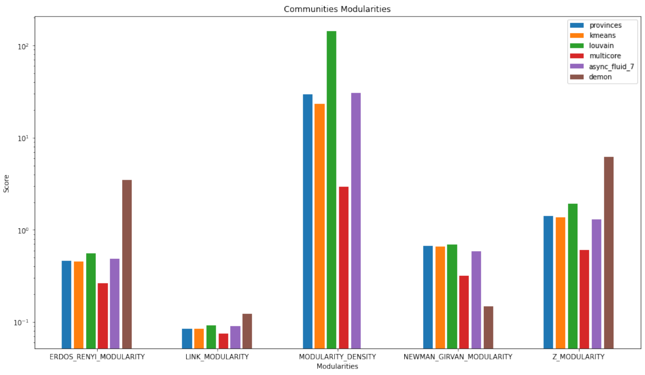


Figure 6: Modularities for the best partitions

- Dimensionality of Communities
- Differentiation w.r.t. Provinces

The first aspect. concern how many communities are there and what is their average size. We can note **demon algorithm found 220** communities and their average size is small. **Louvain with 21 communities** has got some small communities but average size is still high. We can observe also in the scatter plot in figure 7 the inverse correlation between the *average transitivity*[4] of each community and its size; we can also admire the whole multitude of demons communities having a small size and some louvain's too. *However Louvain communities seems to have the best compromises between the two directions.*

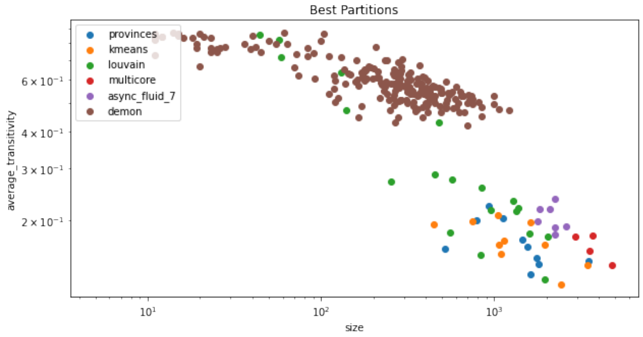
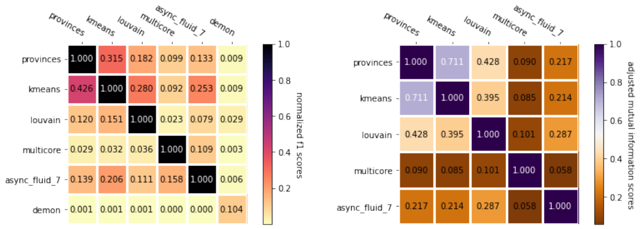


Figure 7: Sizes of the communities and their average transitivity

The second aspect. regards the **normalized f1 score** (*nf1*) and the **adjusted mutual information**. All applied techniques have been compared with provinces using these two metric. While mutual information is symmetric, in the table 4 we have prediction from communities to the provinces. However the opposite is observable from the matrix in figure 8a; seems is easy to predict from kmeans to provinces respect the opposite, and their are really similar. Also *louvain is pretty predictable*; practically single provinces have been splitted in more parts in that case; and the async fluid method has still some similitude. *Multicore and for obvious reason the demon algorithm, lead instead to totally different groups of nodes.*



(a) Normalized f1 Score **(b) Adjusted Mutual Info**

Figure 8: Information Matrices

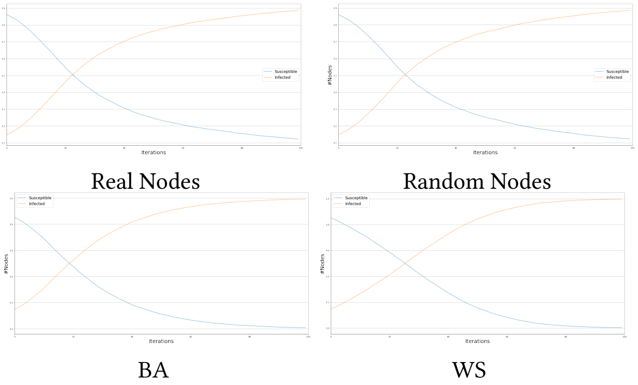
6 DIFFUSION: SPREADING

This segment of our study was dedicated to the study of diffusion models on the obtained and the generated graphs. Our network was composed from nodes representing restaurants, therefore we opted for a different approach to the analysis. We went for the adaptation of special menu's on the restaurants evaluated as diffusion of innovations. Since reviewers often compared options on a restaurants with the options of another, we decided to analyse how the restaurants got the ideas from the competition given the clientele. On special menus we opted for those giving vegan options since it has been a growing trend in recent times. For the analysis we utilized the **NDLIB** [3] utilizing the **SI Model**[5], **Threshold Model**[9], **Kertesz Threshold Model**[11], and **Profile Threshold Model** [8] from it. Parting from a certain number of restaurants that had already adopted the trend, the SI intuition that was used allowed us to observe how its information would spread to other restaurants. The SIS, SIR models could have been interpreted however, the interpretation would have been convoluted so we opted for replacing them with different models. There was a curiosity brewing between us and we decided to evaluate the original network twice, one with the actual nodes that represented the restaurants with the options and the second with nodes selected at random. The *Erdös–Rényi* (ER), *Barabási-Albert*(BA) and *Watts-Strogatz*(WS) models were also tested with a random sample. The results discussed here will show and highlight the most relevant findings.

[4] Average Transitivity is defined as the average clustering coefficient of its nodes w.r.t. their connection within the community itself

SI Model

For the **SI Model** our initial state was determined by the restaurants that already had the vegan options available. We calculated the amount of restaurants that had vegan options on their menu. The total percentage of restaurants reporting vegan options was 14.11%. It was a decent percentage of the population that had listed this kinds of options so we opted to select 14% as our starting population for the random selection for all experiments with random samples. We selected a *beta* value of 0.005 for the infection rate. It was selected after several test in which it was deemed a fitting value for visualizing the spread. As for iterations we found that the most interesting results generally fell inside the first 100 iterations. In the results obtained, depicted in figure 9, we could visualize that the real node selection and the random selection had similar behaviours, both did not achieve total saturation in a 100 iteration mark. However, both had the majority of their development within 40 iteration cycles. This trait was shared amongst all the experiments. The synthetic networks, on the other hand, had a faster exponential growth and reached the complete sample infection inside the 100 iterations range. The Watts-Strogatz had a similar cross-point to the original graph, meanwhile the Erdös–Rényi , Barabási-Albert presented this behavior closer to the 20 iteration mark.



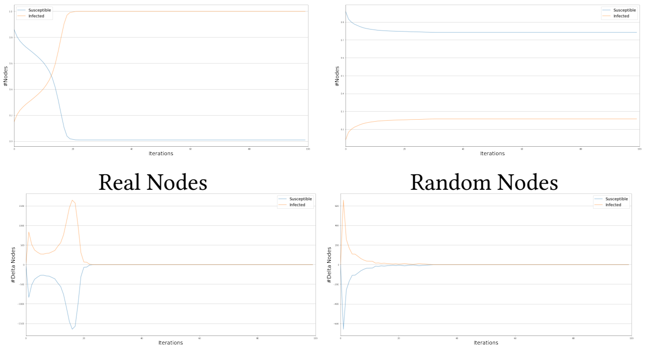
Real Nodes Random Nodes

BA WS

Figure 9: SI Model Diffusion Trend

Standard Threshold Model

The evaluation of the standard **Threshold Model** was made as a look into the pressure that the market imposes into the other restaurants, We understood that a large adoption of a trend in the environment could push the other restaurants to adopt it out of necessity to keep up. For the parameters we chose 35% as the threshold. We saw this as a proper percentage since it has a large enough margin to consider a growing trend relevant and valuable. In the results we found that the Random Nodes selection did not reach the total network. We could understand from this that the network had some deadlocks given the conditions. The ER and BA models showed the same behaviour however they failed to reach a 20% infection while the random selection reached close to 25% as can be seen in figure 10. The real distribution of the nodes reached complete infection of the network close to 20 iterations. The real distribution had a better distribution for the peer pressure approach we took as an understanding for this experiment.



Real Nodes Random Nodes

Real Nodes Diffusion Prevalence Random Nodes Diffusion Prevalence

Figure 10: Standard Threshold Model Results

Kertesz Threshold Model

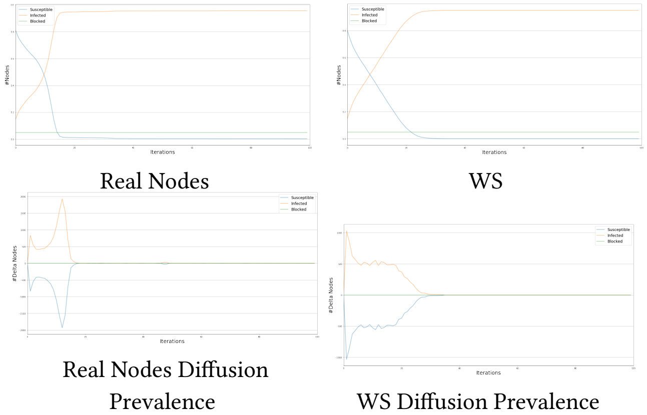
We selected the Kertesz Threshold Model since it extends on the classical model by introducing a density r of *blocked* nodes which are immune to social influence and it also introduces a probability of *spontaneous adoption* p to capture external influence [5]. For this we selected an *adopter rate* of 0.005 and a *blocked section* of 5% as restaurants adamant on ignoring the trend. Given these parameters and the earlier threshold of 35%. We found interesting that even with blocked nodes the inclusion of the adoption rate was enough to give the random selection sufficient extra nodes to trigger the cascade into complete infection [6] similar to the graph of Real nodes found on figure 11. We also found that the WS had a faster and almost linear infection rate in comparison to the others more exponential growth.

Profile Threshold Model

The **Profile Threshold Model** was preferred over the **Profile Model** since we understood that in the restaurant business the trends of the market dictate a part of the business decisions of the restaurants, therefore the peer pressure factor could not be ignored in its totality. By interpreting that

[5] Examples for this case were considered to be among outside information providers like news and blogs

[6] With exeption of the blocked nodes

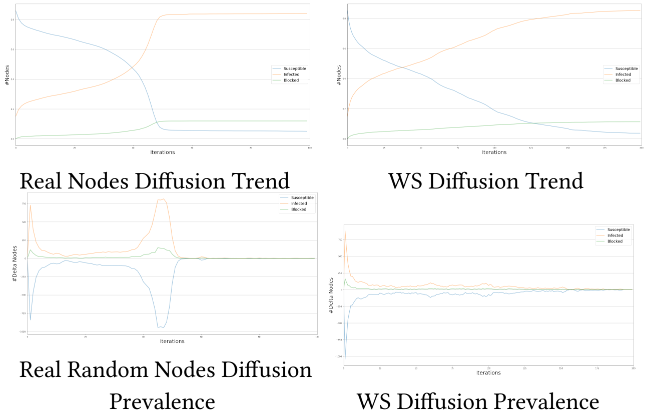


Real Nodes WS

Real Nodes Diffusion Prevalence WS Diffusion Prevalence

Figure 11: Kertesz Threshold Model Results

individual could sour to the new idea and some would out right refuse, this model combines the majority of features and intuitions examined in the earlier models. As depicted in figure 12, we observed how similarly to other experiments, the iterations neded where under a 100 for the Real distribution. However, we note that, in the *WS* was the fist need for more than a 100 iterations to observe the complete behaviour. The *WS* results how it's graph grows slowly and constantly in contrast to the exponential growth experienced by the Real Nodes and the others. In the start seen on their Diffusion Prevalence graphs was evident that they both started with a strong infection. However, real nodes experienced another spike to complete the infection while the WS network only could continuously grow.



Real Nodes Diffusion Trend WS Diffusion Trend

Real Random Nodes Diffusion Prevalence WS Diffusion Prevalence

Figure 12: Profile Threshold Model Results

7 ALGORITHMS: MULTICORE

This section explains a new community discovery algorithm, named **multicore**. The given name derive from the fact that sort of most central nodes called cores are researched, and **communities are at the beginning adjacent nodes belonging to the core of the network**. After that, **communities are propagated by weighted votes**, to the border nodes adjacent to a community and so for, till all nodes belong to a community.

Core Nodes

The core of the network is defined as:

$$Core = \{p : p \in Vertex \land |N_p| >= \alpha \times |V_p|\} \tag{1}$$

...where α is a parameter in $[0, 1]$; V_p is the set of the neighbors of the node p, while N_p is the set of *close neighbors* of node p, or rather :

$$N_p = \{q : q \in V_p \land |D_{pq}| <= \epsilon_p \land |D_{pq}| <= \epsilon_q\} \tag{2}$$

We use D_{pq} to indicate the distance (weight) between nodes p and is adjacent node q.
The **Local Close Neighbor Threshold** ϵ_p of node p is the geometric mean of the distance of node p from all its neighbors and it is defined as follows:

$$\epsilon_p = (\prod_{q \in V_p} D_{pq})^{\frac{1}{|V_p|}} \tag{3}$$

The Multicore Algorithm

In the code snippet 7 are only listed the three phases of the algorithm. The initialization make use of the core notion described. Practically each group of contiguous core nodes, make a different community.
After that, till there are unseen nodes, at each iteration the algorithm consider only the border nodes[7] of communities. Each one of them is assigned to one community, using the **weighted vote** of its neighbors already labeled (assigned to one community). *The weights are given by the* ***centrality of the neighbors****, or rather the distance from the core of the community they belong to.*
Once all nodes belongs to a community, small noise communities, with size smaller the the *minsize parameter*, are deleted and attached to largest communities. This step obviously can be deleted and it will be curious to test the algorithms also without the *"small community deletion"*, while the algorithm evaluated in 5 works removing the noises.

Algorithm 1: Multicore

```
Data: $G, \alpha, minsize$
Result: crisp communities, centralities
initialize communities;
expand communities;
incorporate noise communities in other communities;
```

[7] border nodes are adjacent to at least one node of at least one community

8 LINK PREDICTION

Introduction

Link prediction(LP) is the task of computing the likelihood that a link exists between two given nodes in a network. Two different approaches have been tried in our data. The first one is the classical unsupervised approach used for link prediction by the most of literature. The second approach has been an experiment using supervised learning, with data labeled as connected or not connected.

Unsupervised Learning

For the unsupervised approach we use three different metrics:

- Common Neighbors(CN)
- Jaccard(J)
- Adamic Adar(AA)

On the net used before, cause of the size, the use of these have high computational requirements, so we decided to use a smaller network basing mainly on the time component. In fact we selected from our crawled data only reviews made before 2014; so we reduce drastically the size of the net to 4000 nodes, maintaining also the date in which each edge appeared for the first time. This information was useful to split the network in a temporal manner (as proposed by [4]) ,and so to evaluate the approaches. So we hide from the dataset all the links appeared after a certain date, using them as test set, and we calculated the three metrics on the training set. Highest scores obtained are shown in the table 6 for the Jaccard's coefficient, and in table 7 the Adamic-Adar. What we observe is that almost the top five scores obtained with Jaccard's have the same maximum identical value of 1. Adamic-Adar instead as various results and make possible to better order the possible links.

Link	Score
Mimmi - Pizzeria Alcova del Duca	1.0
Osteria dei Quattro Gatti - Pizzeria La Voglia Matta	1.0
La Battiglia - Ristorante Pizzeria Steakhouse Fuorigioco Campo di Marte	1.0
Osteria Dal Conte - Enoteca Aldobrandesca	1.0
Il Terzo Cerchio - Ristorante Albergo Posta Marcucci	1.0

Table 6: Top-5 links obtained with Jaccard

Link	Score
Marchetti Stefania - Ambaradan Caffe	2.971955558930479
Ristorante Pizzeria Las Vegas - Ambaradan Caffe	2.686087900405103
Il Focolare - La Ruota	2.599443041705373
CM Caffetteria - Panigacceria Taraballa	2.484217772959126
Ristorante Pizzeria La Venezia - El Gallito	2.3147203738875723

Table 7: Top-5 links obtained with Adamic-Adar

Panificio Di Nauta Vincenzo	La Tana Del Ghiro
Ristorante Toscana	Acqua e Sale
Biribo Ristopizza	Ristorante Piccoli Chianti
Trattoria Accadi	Biribo Ristopizza
'Festa del Mugello	Ristorante Toscana
Dogana	Panificio Di Nauta Vincenzo

Table 8: Top-5 most similar nodes obtained with Node2Vec

To evaluate the scores of the three approaches has been necessary to check which are the appearing edges on the whole possible set of link imaginable (so the Cartesian product of the nodes, except for the link of the training set). So by the ordered scores we use **ROC curves** and **AUC** to compare performance. From the roc curves in figure 13, the Adamic-**Adar model has the best performance,** both in terms of AUC and in terms of precision trend, while jaccard's coefficient practically failed in the purpose.

	AUC	Precision
Common Neighbors	0.001317830102250289	0.028525399207496034
Adamic-Adar	0.001487387714381032	0.0285257081380211
Jaccard	0.0009871704625349035	0.028524796752349185

Table 9: Model Evaluation

Supervised Learning

This methodology is actually experimental; in fact creates a **model not able to propose most probable appearing links, but able , given a pair of nodes with their characteristic, to binary guess if the link exist or not**. We start from the giant connected component we have on the backboned network. To create a suitable dataset, the first step is to **create randomly negative samples of unconnected node pairs**, at least enough to make the dataset balanced respect to the target attribute ("connected"). At this point we combine attributes of the two node for each pair, in order to

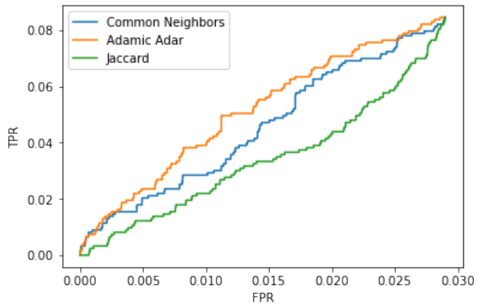


Figure 13: Roc Curve of the unsupervised algorithms

create **new derived numeric features**. You can see a sample of the created dataset in table 10. Categorical attributes such as *"province"* or *"city"*, take original values are equals for the two restaurants. Boolean attributes (*"menu"*, *"covid-Measure"*, etc...) are mapped as logical; numerical attribute are summed or subtracted (latitude and longitude instead give euclidean distance); and lists of values as *"cuisines"* and *"specialDiets"* has been mapped as the cardinality of the intersection of the two sets.
So our dataset was split to obtain a training set composed by the 70% of dataset and the test set composed of the 30% ; and so task was completed using the following supervised algorithms:

- Random Forest(RF)
- Adaboost
- Bagging

Initially the algorithms gave all of them around the 75% of accuracy on the test set. **So we add topological measurement to the dataset: the number of common neighbors and the jaccard's coefficient.**

N.B.*We calculated this two measurement using only the edges from of the training set, creating the relative network and using it to fill the features of both the training and the test set.*

After adding those, the **Random Forest reached an accuracy of 0.86** improving so the performance of ten percent points w.r.t. the previous model. Boosting and Bagging were also applied using the Random Forest as base classifier and reaching almost the same performance with an accuracy of 86% and 75% respectively.

Node2Vec

We are going to compute embeddings with the unsupervised node2vec algorithm. After obtaining embeddings, a binary classifier can be used to predict a link, or not, between any two nodes in the graph. The whole process is divided into 4 main steps:

1. Obtain embeddings for each node
2. Train a classifier For each set of hyperparameters
3. Select the classifier that performs the best
4. Evaluate the selected classifier on unseen data to validate its ability to generalise.

We carefully splitted our dataset to avoid data leakage and evaluate algorithms correctly:

1. a **Training Set** of positive and negative edges that weren't used for computing node embeddings
2. a **Test Set** composed of positive and negative edges not used for neither computing test node embeddings or for classifier training or model selection
3. a **Test Graph** for computing test node embeddings with more edges than the Train Graph, obtained from removing the test set of links from the full graph
4. a **Train Graph** obtained from the Test Graph, used for computing node embeddings

To calculate node embeddings, we use **Node2Vec**.
These embeddings are learned in such a way to ensure that nodes that are close in the graph remain close in the embedding space. Node2Vec first involves running random walks on the graph to obtain our context pairs, and using these to train a Word2Vec model. During Word2Vec training process:

1. We calculate link/edge embeddings for the positive and negative edge samples by applying a binary operator on the embeddings of the source and target nodes of each sampled edge.
2. Given the embeddings of the positive and negative examples, we train a logistic regression classifier to predict a binary value indicating whether an edge between two nodes should exist or not.
3. We evaluate the performance of the link classifier for each of the 4 operators on the training data with node embeddings calculated on the **Train Graph**, and select the best classifier.
4. The best classifier is then used to calculate scores on the test data with node embeddings calculated on the **Test Graph**.

Below we can observe our 4 different operators used to select the best classifier.

Name	ROC AUC score
operator hadamard	0.876742
operator L1	0.924210
operator L2	0.927042
operator avg	0.716665

Table 11: Classifier Evaluation

We can notice that the model that performed better was the operator L2. After selecting our best model, we calculate a final evaluation score, which is ROC AUC score on test set using operator L2: 0.927445451183432.

u	v	province	city	geoDistance	reviewsDiff	scoreDiff	priceDiff	reviewsSum	scoreSum	priceSum	specialDiets	cuisines	menu	covidMeasure	travellersChoice	claimed	connected
7113	4007	1.0	1.0	0.001580	3.0	0.0	0.0	43.0	9.0	2.0	1.0	2.0	False	False	False	False	1
9775	8044	0.0	0.0	1.404040	718.0	0.5	2.5	726.0	8.5	2.5	0.0	0.0	True	False	False	True	0
9029	6184	1.0	0.0	0.426159	59.0	0.0	3.0	71.0	8.0	5.0	0.0	1.0	False	False	False	True	1
8079	1773	0.0	1.0	0.030509	53.0	0.5	0.0	103.0	9.5	5.0	0.0	0.0	False	False	False	True	1
9244	6650	0.0	0.0	1.520406	59.0	0.5	0.0	97.0	8.5	5.0	0.0	1.0	False	False	False	True	1

Table 10: Sample from the training set for unsupervised link prediction

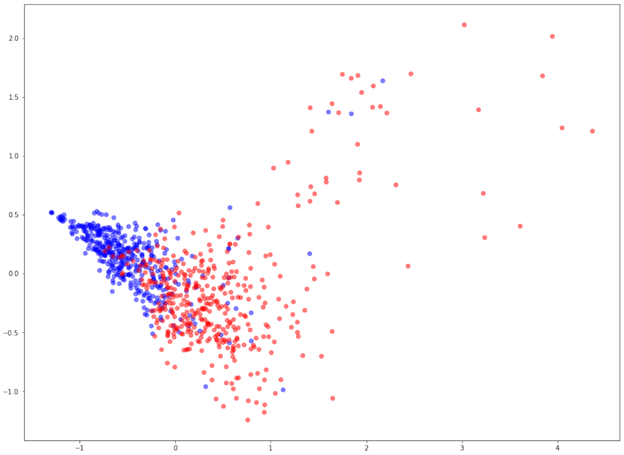


Figure 14: Embeddings visualization

Figure 14 shows a representation of link embedding where blue points represent positive edges and red points represent negative ones. Learned link embeddings have 128 dimensions(the same as the input node embeddings) but for visualization we used the PCA algorithm reducing them to 2 dimensions.

9 OPEN QUESTION

We were reasoning a lot about which kind of research/analysis to make on our dataset. The massive quantity of data make in fact possible to approach them in a lot of different ways, beyond the ones already mentioned. Different network types can be built from our data, and a lot of informations we have but we haven't used yet, can be combined with topological information of the network. Moreover the subjects of the analysis, or rather restaurants and reviews, make us thinking to follow different research environment, such us *cultural*, *economical*, *logistical*, and last but not least , ***epidemiological***.

The true is that there is only one topic that in this moment is a trend and on which to focus on; since the Covid19 break through our lives, our habits completely changed. Italy came across a difficult year and so for was for people and activity. *Lockdown* and *coloured zones* impact in particular coffee and restaurants, pointed out as place of diffusion of the virus. In our analysis we are going to check different aspect of the impact this epidemic has got on the attendance of restaurants, and vice-versa the influence of them on the diffusion of the covid19.

Covid Network by User

To obtain a network, also in this specific case, there are a lot of different manners. For sure it is necessary to filter the reviews basing on the date, but this is not the only one operation done. In fact, since the number of filtered reviews allow us making more complicated processes, the objective is to build a **directed multigraph**:

- *multigraph* because for each different user that reviewed consequentially a pair of restaurants in the period of the pandemic, a new edge appears in the graph between the two respective nodes, named as the user.
- *directed* because having a date for each review, the edge is place from the restaurant first reviewed to the following.

So respect of the previous network, in here not all the restaurant reviewed by almost a same user are connected, but only restaurants, reviewed chronologically below.
In fact, to build the network the process includes multiple action:

(1) the reviews and the relative restaurant have been divided by user;
(2) then for each user, reviews have been divided by date;
(3) dates have been ordered chronologically ;
(4) at this point, for each user, a forward pass through the dates allows to add edges between restaurants of two sequential dates.

At this point, it is just necessary to specify that, in many cases, a user reviewed on the same date more than one restaurant; in this case an edge is created in both direction (so it is like undirected) for all the restaurant reviewed by the user in that date. Moreover, directed edges are created all restaurants reviewed by an user in a same date and all the restaurants reviewed by the same user on the next date. An example fora a random user, is shown in figure 15

The network thus created allow us to track the movement of all reviewer of after 2020. So we can search path and see how hypothetically virus can spread across restaurants. We can see who are the most active users and which are the most visited restaurants. We can also take track over time of what happens ans we have for each edge the date of the starting

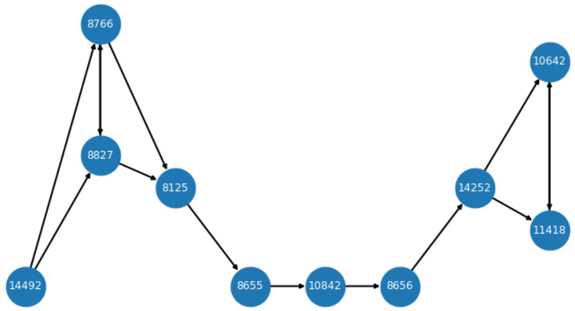


Figure 15: Trace of a sample user

node review, and the date of the arriving node review. Below, an example of the edge attributes for three random edges.

```
(13743, 1076, 0) :
{'start': 194, 'end': 195, 'user': '864matteob', 'geodistance': 0.000386}

(13743, 3848, 0)
{'start': 174, 'end': 180, 'user': 'Alberto B', 'geodistance': 1.04055}

(13743, 3848, 1):
{'start': 176, 'end': 180, 'user': 'moncina74', 'geodistance': 1.04055}
```

The critical issue is that restaurant reviewed by the same user, but not in a direct sequence, are not connected. This can be misleading when calculating the centrality of nodes, or on the path. However it is possible adding edges in a second moment, or better centrality and paths can be weighted using the days spent between the first and the second date.

More Information and External Data

Between the crawled data there are further information regarding the restaurants. In particular we individuated as the most potentially useful for the analysis the geographical information, the province of the node, and the attribute "covidMeasure". We can use **latitude and longitude** to calculate distance between nodes and assign weights to the edges. We can use the **province** to split the network in parts. We can use **covidMeasure** to see if any correlation occurs between this feature and the topology.

In addition to the crawled data, we search on the web for finding **data about the covid trend in Tuscany.** We found what we were searching at the following URL: http://dati.toscana.it/dataset/open-data-covid19. The dataset contains information about the the date, the province of the record, and the **cumulative number of positive people and died people**. So it was perfect to be intersect with our graph, taking in consideration dates and time.

N.B. The first data on the *covidARS* dataset is 12/02/2020 so we use this one as first date on the graph.

From the resulting network is extract the largest weakly connected component. This giant component became so the object of the analysis. It is composed of 9107 nodes and 111741. However considering a lot of edges insist on the same pair of nodes, density is not very high considering the graph and not the multigraph.

Trend Analysis

One the first analysis to come in the mind, is to plot and analyse trends of infections.

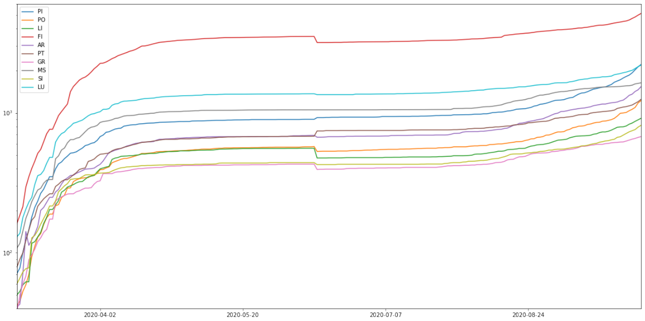


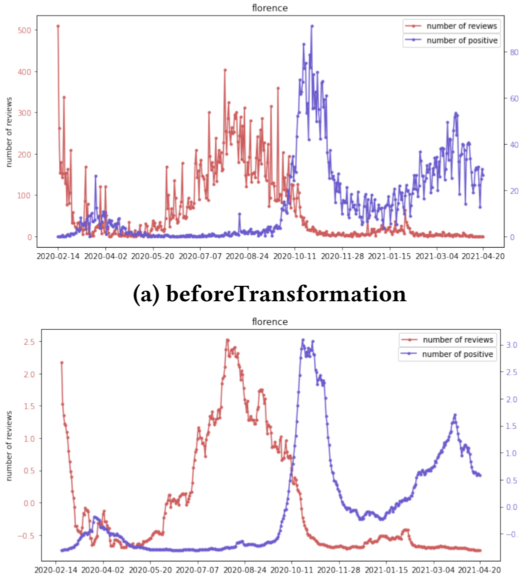
Figure 16: Cumulative number of positive by province

In figure 16 we can see the cumulative lines of infected people province by province[8]. We see there are not too much intersection and as all we know infection started from Florence that always has been the most infected city in Tuscany.
We want to compare these time series, with the one deductable from the network. Actually it is not really topological but make the count of reviews date by date. So edges are assigned to the respective province (looking at the destination node) and we count them. This lines are plotted together with the positives. However till positive were cumulative, we calculate the *"new positive"* (through a lag of the series) time series, better comparable with the number of daily reviews.
It figures out the shape is really similar but lagged in the future for the positive. We can see in figure 17 how the time series appear for the province of ... before and after the transformation we make. In fact to better compare the time series pair to pair, have been applied:

- **Rolling**, with a rolling window of a week to reduce noise
- **Moving Average** To normalize the amplitudes

This transformation allow us to compare the similarity between the two time series. We uses both Euclidean and Manhattan distances, that compare the time series point to point. What we are looking for in this moment, is **to find which is the best lag between the two time series**, understanding how much time the shape of the reviews is ahead respect of

[8] The time series are obtained by transformation on the covidARS dataset.



(a) beforeTransformation

(b) Rolling + Amplitude Scaling

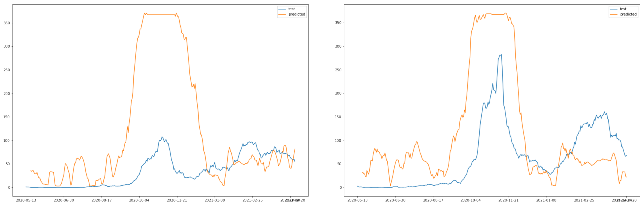
Figure 17: Time Series Transformation

the new positive. So is lagged one date by one, to find which lag give the minimum distance[9]. We made this province by province. On overage both used metrics return **83 days as the best lag**, or rather the lag for which average distance are minimized.

	MAE	RMSE	MAD	R2
Siena	92.409	145.080	33.706	-17.844
Lucca	82.164	108.832	64.802	-1.670

Table 12: Regression Evaluation Metrics

So we provide to lag all time series of positives, in order to build a regression model able to predict positives given the reviews. This time we don’t use the amplitude scaling but only the noise removal. So we use Siena and Lucca as test set and the other provinces as training set. The model we train is a **random forest regressor**, and we train it multiple times for each province of the training, using the ***"warm_start"*** *parameter from scikit-learn* package in python. Then we evaluated the model with different metrics reported in table 12.
In figure 18 the actual time series and the regression are displayed, for the two provinces of the test set.
From the plot and from the metrics we discover prediction in Siena doesn’t go really good; but we should also say that it was the most dissimilar time series from the last analysis on similarity, and so it’s understandable why the prediction is not really correct. While for Lucca the prediction is good after all, also if still not sufficient. In fact we wanted tried to use the model to predict unseen data. In the covidars we have 45 days more than in the network. So we can try to predict them using the last information we have on the network, lagging of 84 days. The obtained prediction are not reliable, but can be a interesting prediction to be interpolated we advanced forecasting techniques.



(a) Siena

(b) Lucca

Figure 18: Actual and predicted on regression

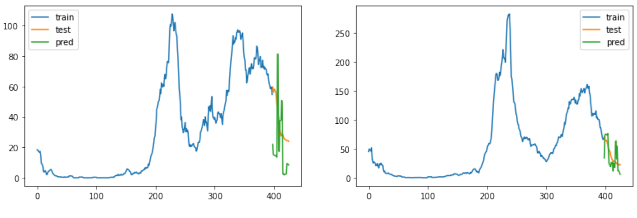
After this attempt, we tried another system, adding more topological information. In fact we use the network edges to build a time series for each province. Iterating over edges, in the point of the time series correspondent to the end date of the edge, we add a value calculated as the new positive per inhabitants in the start date of the edge in the province where the edge is from. The best gap in mean at this point result to be only 28 days. Consequentially the random forest regressor has been trained another time with the newest timeseries, and tested as before; all evaluation measure used have for sure increased[10]. Nevertheless, when we try to predict the unseen data, the model has shown that it is not yet capable of predicting the future on his own. Best result have been obtained on the province of Massacarrara while worst attempt was trying to predict Siena. So we decide to use an **arima model** to predict better data obtaining good result. However by the interpolation of the regression with arima we obtained in media the best results. In the example in figure 19 we plot as example Lucca and Siena prediction.

Users

In this section we tried to analyse the users behavior, and how they moved inside the network of restaurants.
In total our labeled edges shown the existence of 24581 different reviewer. As shown in figure 15 we are able to track each one of them across the restaurants. We wanted also too see which are most active users and which restaurant they visited. So we classify the users by three different points:

[9] the distance has been divided by the length of the time series, cause each "one day lag" we take, the length decrease by one obviously.

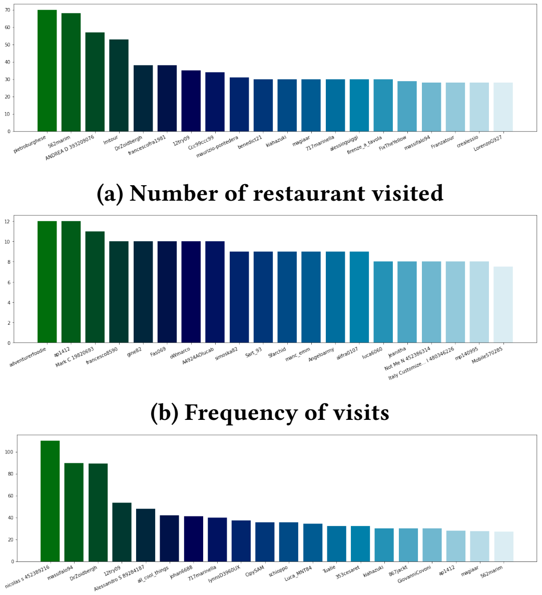
[10] we should remember that in this occasion the train set contains in part also the test set because of the manner in which have been created



(a) Siena **(b) Lucca**

Figure 19: Interpolation Arima - Random Forest regressor

- Number of different restaurants visited
- Frequency: reviews by amount of time
- Traveled distance (aproximated)



(a) Number of restaurant visited

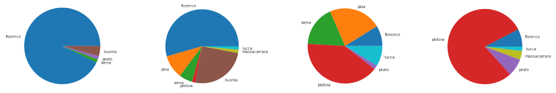
(b) Frequency of visits

(c) Traveled Distance

Figure 20: User

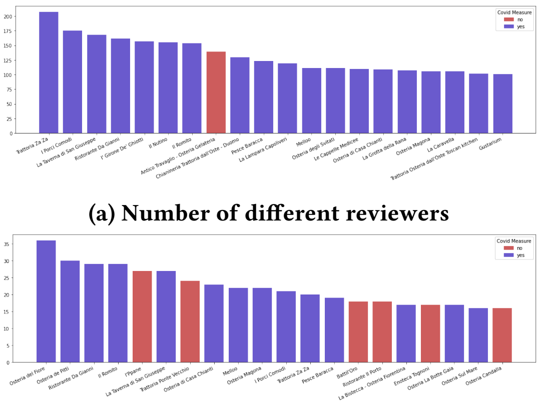
It is visible that a few users in the top of each ranking have considerably higher score than the mediums. We tried to focus on them a little bit more. We discover immediately there is no a restaurant visited by all of them; and also that only the 8% of the restaurant visited by them, holds the *"covid Measures"* target. In opposite of that, we can see form figure 22a that all the most visited restaurant except one have the target.

We focus also the analysis on the provinces they visited. In figure 21 we note "pietroburghese" (that is the one person who has visited the most restaurants) made almost all his reviews on the province of Florence, and so "lmtour" made those in Pistoia. All other top users (included the most frequent and the best traveler) seems to have a favorite province obviously but to move a little bit more around different provinces.



(a) Burghese **(b) marim** **(c) Andrea D** **(d) lmtour**

Figure 21: Provinces visited by the top 4 top users



(a) Number of different reviewers

(b) Number of reviews

Figure 22: Centrality by the number of reviewsreviewers

Sequential Pattern Mining. We continue the analysis on the users, by using the sequel pattern mining. In fact, for each user, is possible to order chronologically by the date, the set of restaurant visited in the same day. In this way a transactional dataset is created (see a sample below).

```
                ...
[(1979,), (1976, 2003), (2163,)],
[(1979,), (1993, 2162)],
[(1979, 2191), (4513, 7245)],
[(2184, 3722, 1979), (1981,)],
[(3293, 14479), (2258,)],
                ...
```

So we applied the prefix span algorithm[10] to see if there is any group of restaurant visited in the same sequence by a consistent group of reviewers. The top results found obviously are singletons, we use their support to plot centrality in figure 22b (6 on 20, or rather the six red bars, have not the covid measure target). The most common pair (support = 17) is

- *"Osteria de Pitti" –>"Osteria del Fiore"* ,

that appears also as the two most frequent restaurants, with supports respectively of 30 and 36. Searching for longer sequence, we found that the most common with length 3 is:

- *"Osteria de Pitti" –> "Trattoria Ponte Vecchio" –> "Osteria del Fiore"*

...with support 9 ; that contains the previous pair just seen; and of length 4 we have two patterns, with the same support of 4 :

- *"Osteria de Pitti" –> "Trattoria Ponte Vecchio" –> "Osteria del Fiore" –> "I'Ppane"* ;
- *"Locanda Fiorentina" –> "I'Ppane" –> "Osteria de Pitti" –> "Osteria del Fiore"* ;

.... the first one is a supersequence of the ones already seen, while the second reverts it a bit.

Paths & Centrality

In this section we concentrated deeper on the details of the net, in its topology and on its geometrical shape. The objective have been to analyse paths and centrality on the directed multi-graph created. Given the way the network was built, make sense to look at paths that respect some time constraints. So starting from Dijkstra algorithm for shortest path, it is possible to modify it, in such a way that time constraints are respected. So in each path dates are always growing; and we can never add to a path, an edge of which the start date is before the last visit date. In this way we calculate and save paths between all pairs of strongly connected nodes. Furthermore, we can use weights in the calculation. So we use the days between the two reviews to weight each edge in order to calculate the fastest paths. Moreover also the geographical distance have been used to weights the calculation. So at the end we used the same algorithm to have three different set of paths:

- Shortest Paths, not weighted
- Shortest geographical distance paths
- Fastest Paths: weighted by the days of the edge

..each one of them can be used to look at the centrality of the nodes , and to discover different informations.

The non weighted paths, allow us discovering the topological diameter of the net created. It has length 45 and regard four different paths, all of them with the same destination: *"Bar Tre Enne"*.

2020-07-23	2020-08-14	2020-08-14
Il Cantuccio	Umami	Ristorante dell'Azienda Agricola Sapereta
Osteria La Torre di Populonia	La Lucciola	Ristorante Bar Maitu
La Baracchina	Ristorante Bar Maitu	Osteria dei Quattro Rioni
Osteria del Faro	Osteria dei Quattro Rioni	Osteria Pepenero
Il Cedrino	Kontiki	Gabbiano 3.0
Osteria del Bastian Contrario	La Rustica	Da Grazia

Table 13: Fastest Shortest Paths

Also have been looking for fastest shortest paths... we found three different path that in the same day pass through 6 restaurants (sometimes the same) .

For the Geographical paths we should say we use the total traveled distance as metric. The Diameter has been found, the length is 14.52 on the euclidean distance longitudelatitude, and it travel across 34 different restaurant(most of them in the province of Florence), and 7 different provinces. It go from "Ristorante Pizzeria Narnali" in Prato, to "Trattoria da Marco Il Crusco" in Viareggio. You can see it in blue in figure 23.

We also checked out for the fastest, as the one that in the fewer time travels more; and we discovered one path from "Cassia Vetus" to "Ozium" that in one day visit five restaurats from five different provinces. Red path in figure 23.

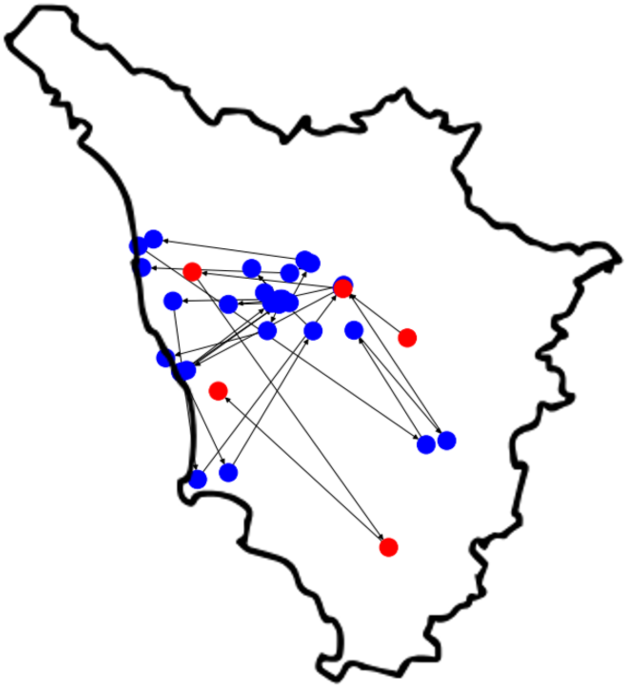


Figure 23: Trace of the geographical diameter

For the fastest paths we note percentage of all paths of have length 0 [11]. However we found also 34 paths with maximal duration of 432 days, all of them starting in the first day of analysis, and terminating in the restaurant *"Miyumi ristorante sushi"* reviewed in last day we crawled data. We check also which is the longest fastest path and we discover that *"Lo Schiaccianoci" –> "Bar Tre Enne"* takes 254 days to pass through 44 restaurants. However 90% of these restaurant are located in the same province of Florence.

We tried also to evaluate Centrality of node using the paths already calculated. The easiest way is to calculate **betweenness centrality**; in fact, with a built-in function is sufficient

[11] In a path with zero days duration, all restaurants have been visited in the same day

to iterate over the discovered shortest paths and take the count of how many of them pass though each node. In figure 24d are plotted the top 20 restaurants for the three different weights we used. The color of the bar indicated whether the restaurant holds the covid measure target or not. We can see on the podium the are always the same three restaurants, in shuffled order. The three restaurants appear also in the figure 22b where sort of degree centrality is shown; so "Trattoria da Zaza" seems to be the most central, the one visited by most unique people, but however is not the most reviewed[12].

We continue using other centrality measures. Connectivity based measure unluckily are long to be implemented; so we decide to use another geometric centrality, the harmonic centrality, that we can calculate always starting from the paths we own. In this case the implementation changes by the kind of path. In fact, we should use different metrics for the distance. By the way, the simple one paths, have the simple length on a graph; for geographical paths we have the weighted paths, while for fastest paths, we just used the total time elapsed as the distance. Results are shown in figure 24h; we see that for normal paths and fastest paths, the harmonic centrality give more or less the same restaurants of the betweenness centrality. Instead harmonic centrality with geographical distances gives really different results. What we see in the relative plot is that we can't find the usual known restaurants (we see only "Trattoria di Zaza" but at the 12th position), wee see new restaurants, most of them without the covid measure target, that are probably tourist restaurant, so people arrive also from other faraway zones of Tuscany. We can conclude that the most central nodes provided to adopt serious measure to covid prevention; but there are still a lot of unprepared restaurants, and are even restaurants visited by tourists and people that comes from faraway .

Provinces Modularities

The last analysis about the movements during the covid, regards the provinces, and how much they communicates. As already done in section 5, here a node partition is created, using the province. So this partition is analysed with the evaluation measure provided by CDLib. All the available fitness measure are reported at table 15 for each province; the modularity score evaluate instead the whole partition, and you can see all the values at table 14 are limited. However we are more interested in the fitness measure, in particular on those of them that regards outgoing edges, such us cut ratio, odf, expansion. For instance we discover the higher values for the cut_ratio and for the avg_odf, in the province of Pistoia. Also the scaled density is quiet higher than the other. This means that respect of the size there are a grate number of edges inside and also a lot of outgoing connection, that contribute moving the virus across the region from a province to another. These measurement can justify way the Pistoia(PT) line of positives in figure 16 is one of the highest, also if it is not one of the most inhabited provinces. Another province with a strange behaviour in figure 16 is Lucca. It is the smallest one (basing on the number of nodes) but is the second more infected province;it has not very high value for the cut ratio or for the avg odf , but seems to have the higher value for the flake odf and for Normalized cut. Hower also the province of Massa-Carrara take high level of positivity in the figure, but doesn't show any abnormal value, they are all in the mean.

We conclude this analysis saying that Florence, the province most affected by epidemic, has not really high value for number of edges and density, and moreover has the highest value for hub dominance; it does mean that also if is infected inside probably doesn't transmit too much people to other province of Tuscany, and that inside of it there are only a few nodes that are the cores of the province.

ERDOS_RENYI_MODULARITY	0.5703064235302439
LINK_MODULARITY	0.03616247289364656
MODULARITY_DENSITY	68.70416792044826
NEWMAN_GIRVAN_MODULARITY	0.10942757495034788
Z_MODULARITY	1.4861854381310924

Table 14: Modularities on Province Partition

REFERENCES

[1] Nesreen K. Ahmed, Nick Duffield, and Liangzhen Xia. 2018. Sampling for Approximate Bipartite Network Projection. (2018). arXiv:1712.08685 [cs.SI]

[2] Michele Coscia and Luca Rossi. 2019. The Impact of Projection and Backboning on Network Topologies. *CoRR* abs/1906.09081 (2019). arXiv:1906.09081 http://arxiv.org/abs/1906.09081

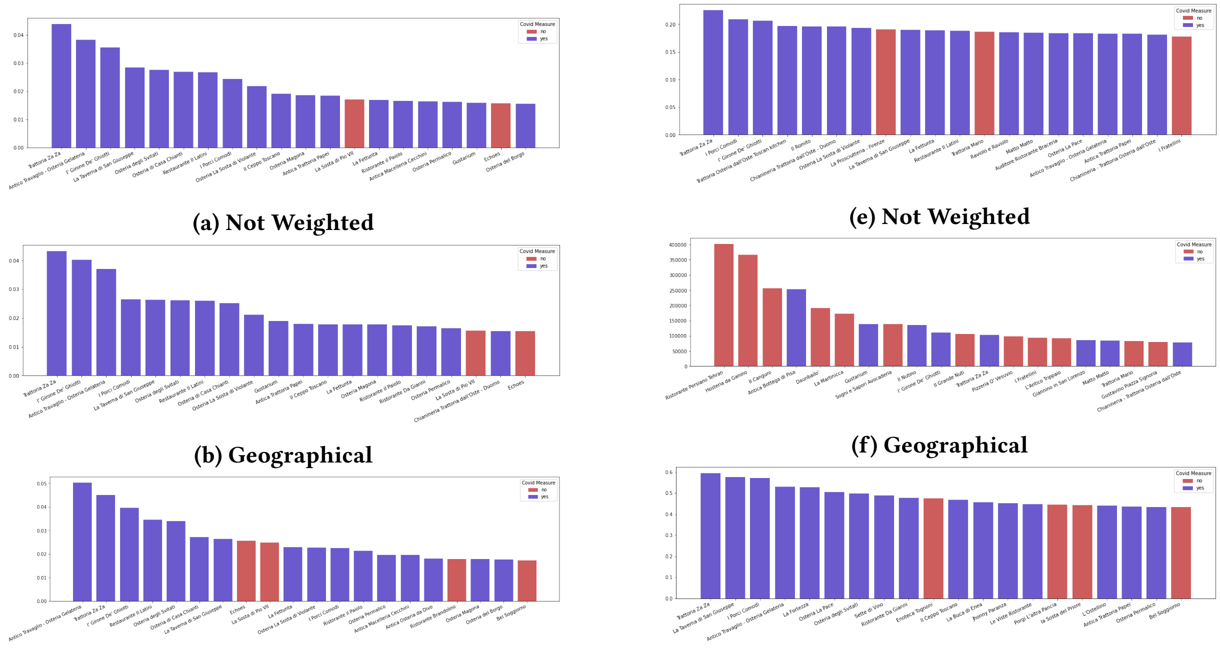
[3] S. Rinzivillo A. Sirbu D. Pedreschi F. Giannotti G. Rossetti, L. Milli. 2017. NDlib: a Python Library to Model and Analyze Diffusion Processes Over Complex Networks. *Journal of Data Science and Analytics.* (2017). https://doi.org/10.1007/s41060-017-0086-6

[4] Zhang Jiawei and Philip Yu. 2019. *Link Prediction.* 229–273. https://doi.org/10.1007/978-3-030-12528-8_7

[5] William Ogilvy Kermack, A. G. McKendrick, and Gilbert Thomas Walker. 1927. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 115, 772 (1927), 700–721. https://doi.org/10.1098/rspa.1927.0118 arXiv:https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.1927.0118

[6] Andrea Lancichinetti, Santo Fortunato, and János Kertész. 2009. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics* 11, 3 (Mar 2009), 033015. https://doi.org/10.1088/1367-2630/11/3/033015

[12] Some people review more time the same restaurant, so the most reviewed is not necessary the one in which most people have been.



(a) Not Weighted

(b) Geographical

(c) Fastest Paths

(d) Betweenness Centrality

(e) Not Weighted

(f) Geographical

(g) Fastest Paths

(h) Harmonic Centrality

Figure 24: Geometric Centrality

	PI	PO	LI	FI	AR	PT	GR	MS	SI	LU
AVG_EMBEDDEDNESS	0.700882	0.743024	0.634341	0.748057	0.520865	0.640908	0.594969	0.492714	0.624210	0.386707
AVERAGE_INTERNAL_DEGREE	20.791919	26.452055	14.132635	21.280510	8.846243	23.597661	8.264014	6.064516	7.293388	4.322807
CONDUCTANCE	0.128000	0.120006	0.190605	0.121975	0.325161	0.180903	0.238079	0.305519	0.241350	0.416943
CUT_RATIO	0.000428	0.000463	0.000418	0.000369	0.000517	0.000632	0.000302	0.000311	0.000269	0.000350
EDGES_INSIDE	20584	17379	8098	11683	3826	10088	2285	1598	1765	616
EXPANSION	3.052020	3.607306	3.328098	2.956284	4.262428	5.211696	2.582278	2.667932	2.320248	3.091228
FRACTION_OVER_MEDIAN_DEGREE	0.493434	0.499239	0.487784	0.468124	0.447399	0.478363	0.435805	0.440228	0.473140	0.494737
HUB_DOMINANCE	0.258211	0.336634	0.253275	0.381951	0.245370	0.442623	0.199275	0.148289	0.165631	0.116197
INTERNAL_EDGE_DENSITY	0.010506	0.020146	0.012343	0.019399	0.010239	0.027632	0.014971	0.011529	0.015100	0.015221
NORMALIZED_CUT	0.160083	0.144507	0.208672	0.137937	0.341957	0.202351	0.244560	0.311861	0.246429	0.420891
MAX_ODF	172	172	193	111	352	225	87	130	68	44
AVG_ODF	6.785354	7.581431	7.386562	6.321494	9.278613	12.238596	5.556962	5.738140	4.942149	6.512281
FLAKE_ODF	0.191919	0.123288	0.247818	0.118397	0.426590	0.233918	0.327306	0.466793	0.278926	0.638596
SCALED_DENSITY	3.898605	7.475756	4.580133	7.198398	3.799317	10.253485	5.555363	4.278296	5.603284	5.648172
SIZE	1980	1314	1146	1098	865	855	553	527	484	285

Table 15: Fitness Evalutation of the Province Partition

[7] Aaron F. McDaid, Derek Greene, and Neil Hurley. 2013. Normalized Mutual Information to evaluate overlapping community finding algorithms. arXiv:1110.2515 [physics.soc-ph]

[8] Letizia Milli, Giulio Rossetti, Dino Pedreschi, and Fosca Giannotti. 2018. Information Diffusion in Complex Networks: The Active/Passive Conundrum. 305–313. https://doi.org/10.1007/978-3-319-72150-7_25

[9] Kermack William Ogilvy and McKendrick A. G. 1978. A Contribution to the Mathematical Theory of Epidemics. *Amer. J. Sociology* 83, Number 6 (1978). https://doi.org/10.1086/226707

[10] Pratik Saraf, R. Sedamkar, and Sheetal Rathi. 2015. PrefixSpan Algorithm for Finding Sequential Pattern with Various Constraints. *International Journal of Applied Information Systems* 9 (06 2015), 37–41. https://doi.org/10.5120/ijais15-451380

[11] Marton Karsai Janos Kertesz Zhongyuan Ruan, Gerardo Iniguez. [n.d.]. *Kertesz Threshold.* https://ndlib.readthedocs.io/en/latest/reference/models/epidemics/KThreshold.html

[12] Tao Zhou, Jie Ren, Matú š Medo, and Yi-Cheng Zhang. 2007. Bipartite network projection and personal recommendation. *Phys. Rev. E* 76 (Oct 2007), 046115. Issue 4. https://doi.org/10.1103/PhysRevE.76.046115