Analyzing the dictionary of the Italian elector

Giovanni Perri, Marco Recchioni, Emanuele Vichi
Github: https://github.com/GiovanniPerri23/SNA_project/tree/main

Abstract

Social media play a key role in shaping citizens' political opinion, in particular Twitter is among the few social networks (if not the only, with millions of users all around the world) that allows its relational data, like the structure of interactions, for the academic use and analysis. Literature on social networks and elections has focused on predicting electoral outcomes rather than on understanding how the discussions between users evolve over time. In the present study we analyse the semantic network observed on Twitter during and after the 2022 Italian political election period and see how elector dictionary changes. Starting from a corpus of tweets posted by Italian users throughout 6 months (August 2022 - February 2023), we apply different algorithms to analyze first the network formed by users and then what is their dictionary, trying to see if there are evident differences also in the semantic that users have during the chosen period. Albeit it is widely acknowledged that Twitter data systematically under-represent the real-world population, there is also evidence that the political system and the role of the incumbent in the election influences the way conversations on Twitter occur.

Keywords: Italian political election; Twitter; Complex networks; Semantic networks; Social network analysis;

I. Introduction

Among all social network, Twitter has become one of the main tools of communication worldwide, including in Italy. It has become recently instrumental for the political functioning of democracies, especially during elections, with the aim to reinforce individualized communication with potential voters and to achieve visibility. It is used by campaigns broadcasting their candidate's messages and interacting directly with possible voters, which according to studies in the UK and the Netherlands has a positive effect on winning more votes [1]. In fact, political conversations about electoral competition and societal policies have been shown to be remarkably more polarized than those about more "ordinary" topics [2].
According to various sites, there are 12.8 million Italian users on Twitter (March 2020), but only 4.8 million of them are active.

For the realization of the project the following environment were used: Python in the Jupyter Environment, with the help of the NetworkX, CDlib and NDlib libraries; Gephi for the visual representation of the graph. For the deepening of the study carried out, we will begin by describing data scraping, followed by a description of the characteristics of the network, specifically degree distribution, connected components, path, clustering coefficient, density and centrality analyses. The same analyzes were also carried out on the synthetic models for comparison purposes.

II. Data collection and preprocessing

A. Tweets collection

The starting point of our analysis is represented by tweets that have been publicly posted from 10 August 2022 to 28 February 2023, divided into three time windows [Agu-Sep], [Nov-Dec] and [Jan-Feb] which will become our 3 datasets. This period was heated by a set of relevant political events, as the Italian political elections (22 September 2022), the handover between the outgoing executive of Mario Draghi and Giorgia Meloni, the Russian gas issue, etc.

Tweets have been retrieved through the library *snscrape*, selecting as key words several Italian political actors from Fratelli d'Italia (FdI), Lega, Azione, Movimento 5 Stelle (M5S), Partito Democratico (PD), and PiùEuropa. The selection of the respective politicians was based on their role in their party and their activity on Twitter (i.e. the most active users for each party were preferred). As results, we download twitter containing at least one of the following words: *'bonino', '+europa', 'meloni','fratelliditalia','calenda', 'salvini','lega', 'azione', 'letta','partitodemocratico', 'renzi','m5s', 'conte'*.

The data acquisition procedure led to a data set of approximately $\sim$50000 tweets, posted by $\sim$2500 users for each dataset .

III. Network characterization

Using the Networkx library, we first built a multigraph, directed and weighted. This structure is justified by the fact that same pairs of users can have more interactions with each other; The idea of a directed graph is justified also by the interactions, which are indeed direct as a user could mention another one but the latter may not would have any kind of connections with the former. However for computational simplicity many algorithms of the Networkx library are set up for undirect graphs, so we decided to switch to an undirected one and work with that. Therefore we obtained a graph having 30350 nodes, 48813 edges and 17 self-loops (users who mention their own tweet).
To better understand the characteristics of the real network, we chose to compare it with four synthetic models, which have been created with features (number of nodes and edges) as similar as possible to those of the real network, by setting the following parameters:

- **Barabasi-Albert model (BA)** $m = 2$;
- **Erdos-Renyi model (ER)** $p = 6 * 10^{-5}$;
- **Watts–Strogatz model (WS)** $k = 4$ and $p = 0.01$
- **Configuration Model (CM)** setting for each network node the degree of the corresponding node in the real network.

A. Degree distribution

Degree distribution analysis is a way of characterizing the connectivity patterns in a network. In our case, just by looking at the degree distribution in fig:1 it is possible to assert that our network was quite similar to the Barabási-Albert model, while it appeared very different from the Erdös–Rényi and the Watts-Strogatz (fig:2). The observed graph followed a power law distribution, as could be expected from a real network. In fact, since gamma is included between 2 and 3 (in particular we obtained a coefficient $\gamma = 2.37$, it is possible to collocate such graph in the ultrasmall world in a scale-free regime.

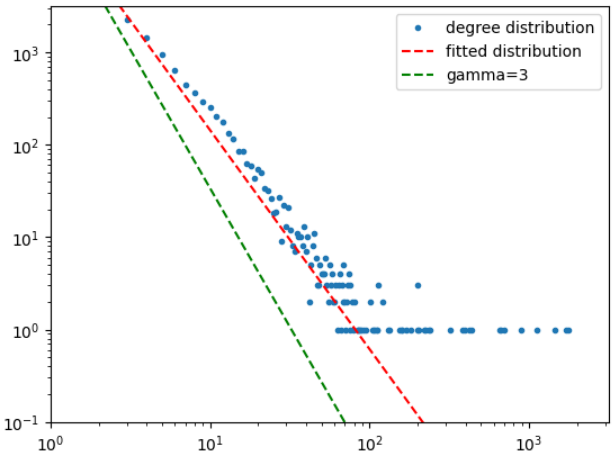

Fig. 1: Fitted degree distribution.

B. ER model

Since $\langle k \rangle > 1$ and $p > p_c = 1/N$, we are in a supercritical regime, in which there is only a unique giant component and the cluster size distribution is exponential. However, it's not surprising that models produce different distributions. The reason behind this difference lies in the fact that the Erdos-Renyi model is a random network model that does not take into account the degree distribution or preference of nodes, while our real network follows a power law distribution, reflecting the preference of nodes to link to high-degree nodes.

C. BA model

The comparison with the Barabasi-Albert model suggests that this model is a suitable one for mimicking the behavior of our network, which exhibit a power-law degree distribution. The model captures the preferential attachment behavior of nodes, which is a common characteristic of real-world networks. Therefore, the Barabasi-Albert model can be used to simulate and study the behavior of real-world networks. In figure 2 the degree distribution plots.

D. CM model

This configuration model is a method for generating random networks from a given degree sequence. We can see from the figure 3 , how the behavior of the synthetic network, specially created having the same degrees as the real network, can in fact be a good representation of our graph.

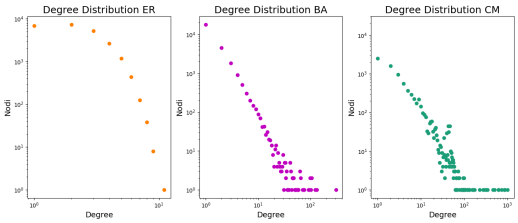

Fig. 2: Degree distribution of the three syntetic models.

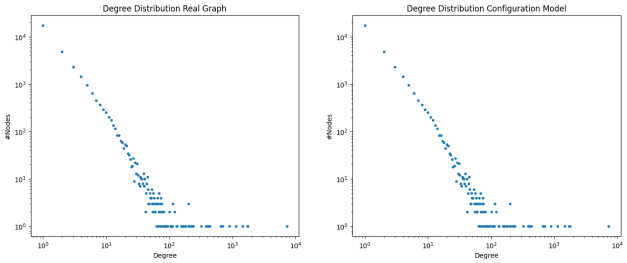

Fig. 3: Degree distribution of real network and CM

E. Other analysis

We carried out an analysis on the connected components of the real graph and of the relative synthetic models, obtaining 1413 connected components for the RW (real world) graph. In particular, another graph was carried out, this time directed and weighted (with the weights obtained through the number of tweets in which the users were mentioned), obtaining a much higher number of connected components, 19501 for the weakly connected and 25371 for the strongly connected. Although we expected at least the same order of magnitude, it is possible, as in our case, to obtain many more connected components in the case of the directed graph. A possible explanation could be that in the case of the weakly and strongly components python calculation, many more nodes are taken into consideration, in particular those with very low degree, which number in free scale is known to be large.

The average degree obtained for our graph is 3.75, and represents the average number of connections that each node has with the other nodes of the graph (i.e. on average, each node of the graph is connected to about 3.7 other nodes). As regards the average clustering and density coefficients, we obtain two values of 0.05 and 0.00012 respectively. The value obtained for the average clustering coefficient is quite low, which suggests that the nodes of the graph are less likely to form clusters or groups. Similarly the density, which represents the fraction of edges present in the graph with respect to the total number of possible edges, is also quite low, therefore the graph has few edges present and therefore is less connected.
Other analyses, concerning the centrality of the nodes, are reported in the github repository.

IV. Task 1: Community Discovery

Undoubtedly an interesting task is the Community discovery(CD), even though it is an ill posed problem, indeed each algorithm models different properties of communities. For each of the following families we select an algorithm from cdlib:

- **Internal density**: Leiden
- **Local-first**: DEMON
- **Percolation**: Label propagation
- **Approach based on random walks**: Walktrap

A. *Preliminar analysis*

Given the structure of the Network, the results of these algorithms share, somehow, the same structure: a few highly dense communities and much more less dense communities. This carry us to decide to put, where it was not already implemented in the algorithm (like DEMON), a minimum threshold for the communities size. So we made an analysis for each algorithms on node coverage as the minimum size varies, the results are reported in Fig:4. As we can see only the Leiden algorithms manages to have an acceptable node coverage as min_size increases, for the remaining three we see an exponential decay as it increases.
For this reason the semantic analysis presented in section IV-D, is computed only on the community detected with Leiden.
Anyway we perform some basic analysis on all the selected algorithm, dividing them among those who generate a clustering partition or clustering overlapping.

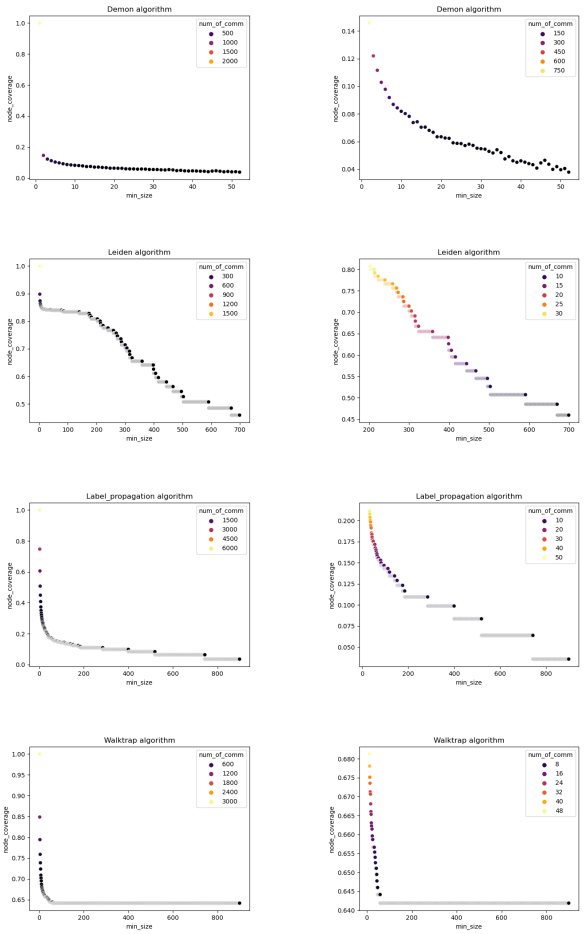

Fig. 4: Results for each algorithm

B. *Crisp Communities*

A clustering is said to be a partition if each node belongs to one and only one community, the $Leiden$, $Label\_propagation$ and $walktrap$ fall in this family.

We start by computing the main partition quality function and community characterization. This internal evaluation is done with the following functions implemented in **cdlib** library:

- Number of communities(1)
- Average internal degree(2)
- Hub dominance(3)
- Conductance(4)
- Modularity(5)

As reported in Table:I.
Then an external evaluation is performed between each pair of algorithm using a normalized mutual information score, the results are reported in Table:II.

	1	2	3	4	5
$Leiden$	1748	1.1	0.96	0.01	0.62
$Label\_propagation$	6241	1.3	0.98	0.38	0.44
$Walktrap$	3067	1.2	0.96	0.12	0.24

TABLE I: Internal stats for each algorithm

	$Leiden$	$Label\_prpagation$	$walktrap$
$Leiden$	1	0.66	61
$Label\_propagation$	0.66	1	59
$Walktrap$	61	59	1

TABLE II: Mutual information

We also try to visualize the communities graph Fig:5-6-7, but obviously given the very high number of communities for each algorithm the total graph is unintelligible although we can observe a central region where there are the denser communities with a ring of lower density communities. So we set for each algorithm a value for the top_k parameters in order to zoom on the denser communities.

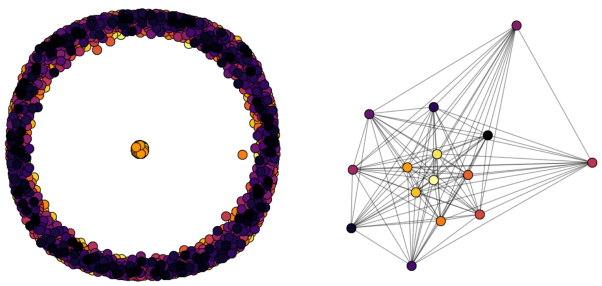

Fig. 5: Leiden algorithm and top_{15}.

Doing this we found that the structure of Leiden and Label_propagation are similar(Fig:5-6) whit the communities that are strictly connected, while for Walktrap (Fig:7) we see that there is a central community connected to the others and only few edges between them.

To conclude the analysis we use LFR Synthetic graphs as benchmark[8], that account for the heterogeneity in

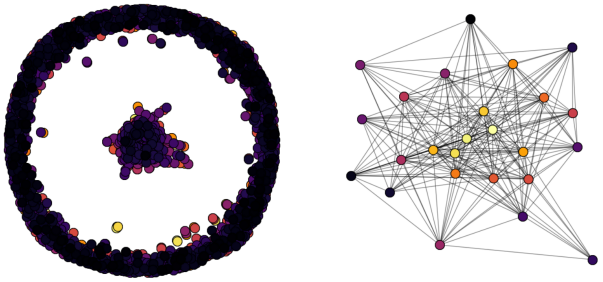

Fig. 6: Label propagation and top_{25}.

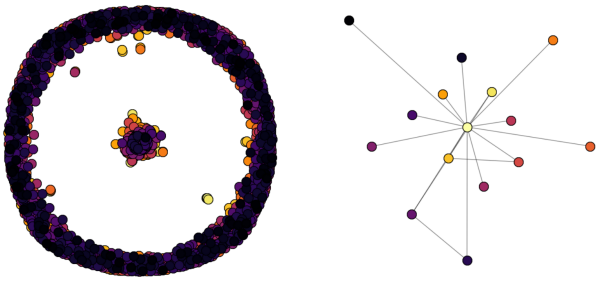

Fig. 7: Walktrap and top_{15}.

the distributions of node degrees, community sizes and assume a power laws distribution for both (which is true for both Label_propagation and Walktrap, as we have seen before, but not for Leiden). Then we compare the synthetic graph with the ones given by our algorithms using $normalized\ mutual\ information$. So we set the parameters of LFR using the information about our network and then leaving the $mixing\ parameter\ \mu$ free to vary, the results are reported in Fig:8.
Lower values of μ represents higher inter-community connection between nodes. In this case the algorithm that better fit the synthetic graph are Leiden and Label propagation hence these algorithms find communities strongly internally connected rather than externally.

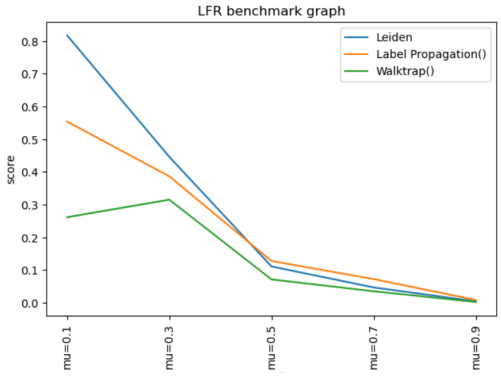

Fig. 8: LFR for ground truth evaluation

So we can observe that the scale-free nature of the original network also occurs in CD, indeed all three algorithms have an high $hub\ dominance$ and the size distribution of the communities follow an scale-free too. In Table:I we can see that that the different nature of the algorithms is noticeable in the $conductance$ and $modularity$ score, the first measures how communities interact, while the second define the communities strength.
For $modularity$ we could expect that Leiden will have the higher because it is an optimization of Louvain algorithm that is a **internal density** based algorithm that try to maximize modularity itself. While for conductance is reasonable that Label propagation has the higher score, because it belongs to **percolation** family algorithm which exploits the propagation of a same property/information to detect community. However the very high number of communities doesn't help us in the analysis and make the internal results unintelligible.

C. Overlapping communities

A clustering is said to be overlapping if any generic node can be assigned to more than one community. We choose from this family a local-first discovery method, the DEMON algorithm, that stand for **D**emocratic **E**stimate of the **M**odular **O**rganization of a **N**etwork [7]. So we run the algorithm setting the parameters: $min\_com\_size = 3$ and $\epsilon = 0.4$. The internal evaluation are reported in Table: Due

	$DEMON$
$\#\ of\ comm.$	320
$node\ coverage$	0.12
avg_int_degree	4.3
$modularity$	0.04
$hub\ dominance$	0.82
$conductance$	0.86

TABLE III: Internal stats for DEMON

to the overlapping nature of the communities, we cannot obtain the $normalized\ mutual\ information$ in respect to the algorithms used before. Then we try to visualize the total communities graph, but unlike in Sec:IV-B there is no general structure to highlight, due to the fact that it has 12% of node coverage and so it detect only the highly connected region without the "ring" around like in Fig:5-6. So we plot the $top_k = 20$ communities only (Fig:9) but setting **True** the $plot_{overlaps}$ parameter.

D. Discussion & Semantic analysis

Considering the semantic information stored in the network's edges (i.e. the actual tweet through which the 2 users interact), we carried out a qualitative word-based inference of the resulting clusters.
In the previous analysis we used different CD approaches, due to the scale-free nature of the network all the algorithm struggle to find well-separated communities. Hence we want to use the algorithm that manages to keep an acceptable node coverage even when raising the min_size threshold of the communities, so we see in Fig:4 that the only one who succeeds in this is the $Leiden$ algorithm.
Therefore using the $Leiden$, we print out the 20 most common words present in the links within each community. For our political dictionary purpose is interesting to show only the first 8 communities (which provide a node coverage of $\sim 50\%$).

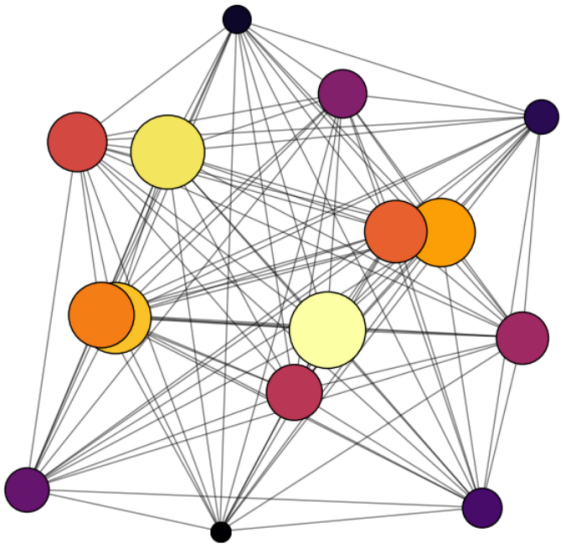

Fig. 9: Visual communities graph for DEMON

Comm. 0	Comm. 1	Comm. 2	Comm. 3
azioneit 10498	fratellilitalia 6268	lega 2749	enricoletta 2612
italiaviva 9763	giorgiameloni 5334	borghiclaudio 1738	pdnetwork 2246
carlocalenda 9399	isabellarauti 662	salvini 932	partitodemocratico 1161
matteorenzi 6027	meloni 650	alexbazzaro 784	letta 1051
meb 5548	fdiparlamento 469	dottorbarbieri 734	bonino 792
elenabonetti 4398	solo 426	lucabattanta 727	emmabonino 697
lucianonobili 3086	fdimilanoprov 396	albertobagnai 699	piueuropa 658
renzi 2534	lombardiafdi 393	matteosalvinimi 680	deputatipd 594
marattin 2486	fatto 371	braveheartmmt 611	pd 570
teresabellanova 2334	mai 358	lucafantuzzi 561	senatoripd 553
pdnetwork 1997	fratelliroma 352	antonelloang 559	meloni 459
calenda 1953	guidocrosetto 344	martinoloiacono 552	annaascani 436
raffaellapaita 1913	governo 329	giuseppepalma78 550	peppeprovenzano 420
maracarfagna 1835	cosa 323	jimmomo 550	cottarellicpi 397
ettorerosato 1813	poi 321	giuslit 550	cosa 396
ivanscalfarotto 1679	fare 314	bufaloema 549	itinagli 393
tnannicini 1636	essere 312	antoniosocci1 549	renzi 365
mrcfasolo63 1629	no 304	xenio 548	sbonaccini 362
claudiafusani 1627	anni 301	federicaf33 548	solo 361
bobogiac 1424	fa 289	ladydd69 548	calenda 357

TABLE IV: 20 most common word for comm.0-3

Comm. 4	Comm. 5	Comm. 6	Comm. 7
conte 1013	matteosalvinimi 906	piueuropa 480	pd 156
m5s 950	legasalvini 737	fratellilitalia 291	renzi 144
giuseppeconteit 853	fratellilitalia 470	pdnetwork 258	calenda 142
mov5stelle 518	giorgiameloni 390	riccardomagi 252	bonino 135
renzi 512	lega 390	sinistrait 237	m5s 130
fatto 321	berlusconi 322	bonino 190	conte 124
letta 294	enricoletta 305	enricoletta 186	letta 119
pd 287	legasenato 294	mov5stelle 185	meloni 105
calenda 282	legacamera 284	europaverdeit 176	salvini 104
draghi 270	salvini 283	unionepopolare 154	lega 98
solo 269	pdnetwork 282	marcocappato 152	carlocalenda 89
fattoquotidiano 245	lntoscana 277	legasalvini 149	lorepregliasco 87
meloni 243	liberoofficial 242	pd 140	draghi 76
salvini 220	matteorenzi 226	m5s 138	solo 71
governo 205	manfredipotenti 217	così 136	fatto 71
fare 184	storace 210	casadileo1 133	governo 59
stato 180	forzaitalia 195	antigoneonlus 131	poi 57
lega 178	quirinale 155	ageofvirgo 130	quando 55
via 174	noimoderati 154	nn 129	no 54
poi 171	credo 153	possibileit 129	cosa 54

TABLE V: 20 most common word for comm.4-7

V. Task 2: Opinion Dynamics

In order to investigate the opinion dynamics of our network, 6 most common opinion dynamics algorithm from Ndlib have been used: Voter, QVoter, Majority Rule, Sznajd, Algorithmic Bias and Cognitive Opinion Dynamics.
This work has been performed on 3 different sets of initial condition for the ”infected” node: at random, using the best hubs of the network and through the community detection.

A. Random and Hubs

The initial percentage of infected nodes is investigated for all the algorithms. For the hubs, nodes are sorted based on their degree and bigger and bigger percentages are taken into account.
Synthetic BA and ER graphs with same number of nodes and (almost) links have been constructed and compared to our network.

Voter and Sznajd
As depicted in the left figure in 10 the Voter algorithm display no change in trend for all models analysed which remain stable for all values of initial percentage with a small exception at $\% = 0.5$ where the variance slightly increases.
As a counterpart the Sznajd alg. display a more interesting behaviour:

- the synthetic BA model results more stable than the true network
- targeting the hubs tends to increase the final outcome of infected even though is not so relevant
- exponential networks maintain linear relation between the initial and final percentage of infected
- for the ER graph there is a phase transition at $\% \sim 0.3$ before which the epidemic dies and after which the epidemic converges to all infected

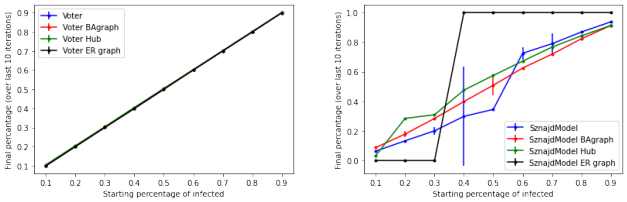

Fig. 10: Voter-Sznajd varying the initial infected nodes percentage

QVoter and MajorityRule
Both of those algorithms as well as the Voter depict a linear trend between initial and final percentage of infected node as the Voter model, hence the graph is not presented here.
Setting the percentage at 0.3 the Q value has been inspected.
Qvoter has basically a constant trend however from 11 (left) is clear how the ER graph is more stable for all Q-values while the exponential networks depict a more variable outcome. The Hubs start results shows slightly higher values than the random one.
Instead the Majority Rule model has the same behaviour for all networks but as Q increases the epidemic (opinion) decreases but with larger and larger variance.

Algorithmic Bias and Cognitive Opinion Dynamics
For this two algorithms the initial node status is set at random, exception made for the Hubs task where a

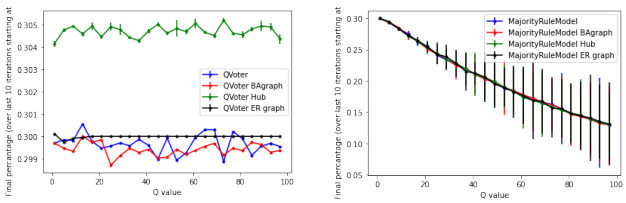

Fig. 11: QVoter-MajorityRule varying the Q parameter

percentage of main hubs are set to a high value status: 0.9. (That was done by a slight change in the source code of the algorithms).
From a straightforward application the opinion evolution of those models present different behaviours as shown in12.

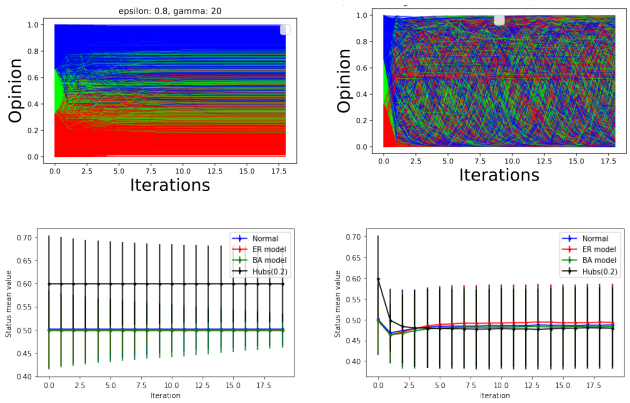

Fig. 12: OpEvolution and mean value for AlgorithmicBias(left) and CognitiveModel(right)

Those graph were taken setting fixed random values of the hyperparameter to give an insight of their trends. The opinion evolution graph has been done on the original Network while a total mean value of the status is plotted for every graph type.
The AlgorithmicBias tends to slightly mix up the opinion on the borders of the 3 ranges. However the overall mean status remains constant for all iterations but the variance decreases. (An higher mean for the Hubs model is because 0.2% of the hubs is set with a high status.)
The CognitiveOpinionDynamics parameter which initialize the node are set to range free $[0, 1]$ and the external information parameter is set to 0.3.
This algorithm shows a total mixture of the starting opinion but an overall mean which converge to 0.5 even after infecting the Hubs.

To investigate how the Hubs change the outcome of the two algorithms different plot of the mean values are drawn with different percentage of hubs infected as shown in 13. Once again it is clear the stability of the AlgorithmicBias which maintain the initial mean value status in opposition at the CognitivModelDynamics which converge to 0.5 for all starting point.

At last, a plot of the difference between the final and initial status for different parameters values is performed

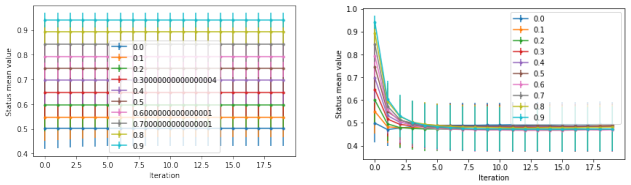

Fig. 13: Mean status values varying the percentage of infected hubs

to give an insight on how those parameters effect the algorithms.
AlgorithmicBias has two parameters to tune, gamma and epsilon. Epsilon representing the open mindness of a node to change its status and gamma the repulsion to interact with far away status nodes.
Setting $gamma = 50$, the delta status remain constant at 0 for all epsilon values tried and for all kind of networks meaning that no change from the start to the end has occurred. Same behaviour as when the gamma parameter is change to respect to $epsilo = 0.2$. (Since those graph were no interesting they are not shown here).

For CognitiveOpinionDynamics an inspection on the effect of the External Information parameter has been done as in 14.
The starting point when set at random is 0.5 (0.6 for the Hubs task, and since it converges anyway to the other model, the down-shift is consistent). Negative values in this graph depict a convergences to lower status points. This behaviour is linear to respect to the parameter for which at 0.4 the model exhibit a convergence towards the 0.5 status, which rises with the further increase of the parameter.

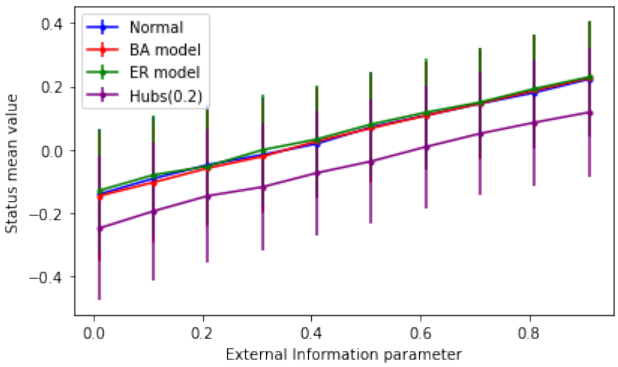

Fig. 14: Delta status vs External information parameter

B. *Using Community*

In this case the communities found (with Leiden algorithm) in the previous section have been use to set the initial conditions. The most common words representing the right and left wing of the political parties found in the communities were used to infer the status of such nodes (without loss of generality the nodes in the

communities with right parties words were set as infected).

Voter and Sznajd

While the Voter model is stable due to the small percentage of the initial "infected" nodes, the Sznajd model shows some fluctuations around the initial percentage as can be easily noted in a direct comparison of the two in 15.

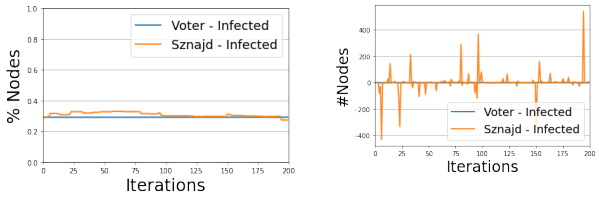

Fig. 15: TrendComparison and PrevalenceComparison Voter-Sznajd

The stability of this dynamics can be addressed either to the nature of the algorithms, the low percentage of infected nodes and due the fact that those infected belong to communities which are more likely to interact with itself by definition.

QVoter and MajorityRule

Once again a discrete state depict the infected and the Q hyperparameter for this two model reflect the number of neighbours to take into account. To see how this parameter affect the trend of the algorithms a plot for different Q values compared to the number of susceptible nodes at the end of the simulation have been carried out as depicted in 16.

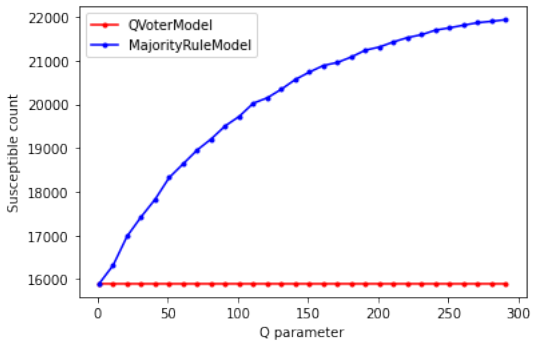

Fig. 16: Susceptible count vs Q parameter for QVoter and MajorityRule algorithm

Since the right wing eventually won the election a small Q (15) have been used. In (17) a comparison of the two algorithms.

Algorithmic Bias and Cognitive Opinion Dynamics

A continuous status $[0, 1]$ of the node reflect its opinion.
In this case many different hyperparameters can be set for the two.
In order to set the node status at our will a slight change on the source code have been performed. The nodes in the communities with right or left political words

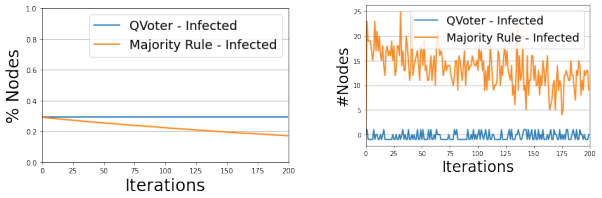

Fig. 17: TrendComparison and PrevalenceComparison QVoter-MajorityRule

have been set in a random state between $[0.66 - 1]$ and $[0 - 0.33]$ respectively. While nodes in minor communities or ambiguous one (with many right and left political words) are set at random in a range $[0.33 - 0.66]$.
The OpinionEvolution visualization method don't really give the chance to depict the opinion trend due to the large number of nodes(example on 18).
A plot of the main value of the status (19) of the nodes

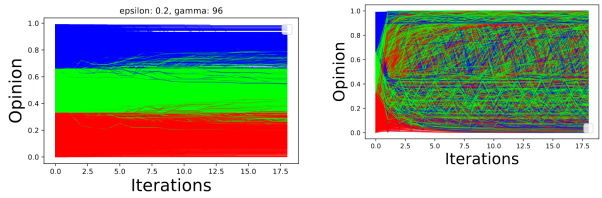

Fig. 18: OpinionEvaluation plot for AlgorithmicBias and CognitiveOpinionDynamics model, pre-parameter tuning

belonging to the three ranges described above have been done to give an insight of the trend of this three regions.
The AlgorithmicBias model is stable while the

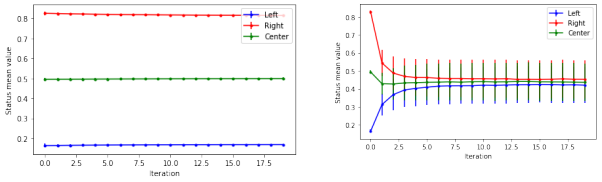

Fig. 19: Left: AlgorithmicBias model pre- parameter tuning
Right: CognitiveOpinionDynamics pre- prarameter tuning

CognitiveOpinionDynamics shows a mix up in the opinion of the three regions (that can be seen as the variance of the status highly increases) and a general convergence toward a particular status.
Since the election were won by the right political party the hyperparameter tuning is performed to seek an opinion trend towards high status values.

As before, graph with the deltas of the final and initial mean status value has been done varying the hyperparameters.
The AlgorithmicBias Model maintain its stability even during the epsilon and gamma parameters tuning (20)
The mean values of the status remain in the starting region and no general mix-up appear to happen for all

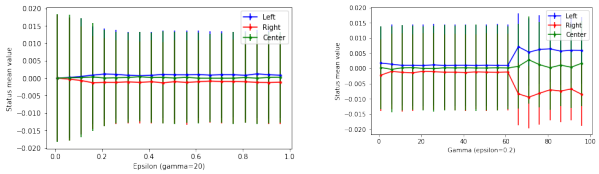

Fig. 20: AlgorithmicBias prarameters tuning

the parameters values tried.

For the CognitiveOpinionDynamics model many are the hyperparameter to tune (risk sensitivity, tendency to inform others, trust in institutions and trust in peers) which set a limit in the random parameters describing the interaction of a node. Since no prior information are available no limits were imposed on those hyperparameters. Instead a tune of the External information parameter has been performed.
As can be seen in 21, increasing this values shift the absolute increase of the status mean to higher values. In order to simulate the election a value of $I = 0.9$ has been set showing a convergence towards a 0.7 mean status values.

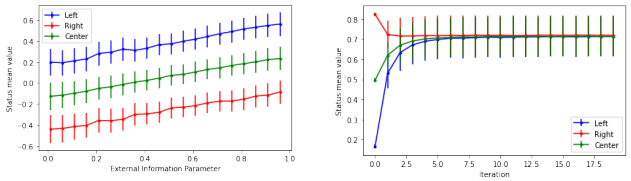

Fig. 21: CognitiveOpionionDynamics prarameters tuning and final plot

VI. Task 3: Link prediction

Another useful task on a political content network is link prediction. The problem is, as Nowell and Kleinberg say, *given a snapshot of a social network, can we infer which new interactions among its members are likely to occur in the near future?* We used approaches [5] based on measures for analyzing the "proximity" of nodes in a network, trying to predict the likelihood of future interactions between users, based on their historical patterns of mentions and interactions. Otherwise, link prediction might also be seen as the way of predicting the links that the data crawling methodology failed to capture.

This task had a high computational complexity since the set of possible edges to be predicted is $O(V^2)$ where V is the number of nodes [6]. For this reason, it was necessary to implement the selected techniques on a subsample of the giant connected component. To do this, only nodes with degree greater than 40 were selected, obtained a subgraph of 300 nodes for the training set. For the test test, instead of using the same dataset and divide it with the classic partition 80-20, we decided to use tweets published between November 1st and December 31st, immediately after the election. We finally create our subgraph by selecting only nodes with degree greater than 50, obtaining about 150 nodes and 2700 edges.

A. Unsupervised algorithms

We applied some unsupervised algorithms, such as **Jaccard**, **CommonNeighbours**, **AdamicAdar**, **Katz** and **SimRank**, that broadly make the assumption that nodes with similar network structure are more likely to form a link. All of them first exclude edges already present, to predict only new links; they differ from each other in how they compute the distance between nodes and such measures are reported for completeness in the table 2. Some algorithms such SimRank and Jaccard did not bring much additional information (we could infer only some useless link prediction between ordinary users - i.e. not verified account). One possible reason could be that they focus primarily on common node counts, while the other three evaluate node similarity based on more complex factors, including the position of these nodes in the network, and their relative importance. Indeed they provide more relevant information, such the link prevision between the main politicians and deputies. We applied these predictors to the training network and then we evaluated the obtained prediction against the second dataset. Results of the predictions' accuracy are shown in the figures 22: although isomorphic, both ROC (with associated AUC, in Table VI) and precision recall curve were reported.

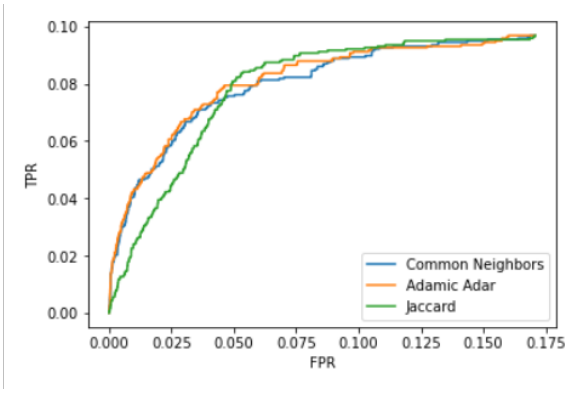

Fig. 22: ROC curve

Raising the threshold on the degrees of nodes, we noticed that all 3 predictors gave better results, albeit slightly.

TABLE VI: Area Under Roc Curve (AUROC)

Algorithm	Score
Common Neigh.	0.0136
Adamic Adar	0.0137
Jaccard	0.0133

VII. Open question

The last phase of our analysis focused on analyzing the vocabulary used by voters, in particular how it evolved over time, starting one month before the election and ending in February 2023. We used 3 temporal slot as said

in subsec:II-A thus obtaining 3 different datasets which we will call from now on respectively *dataset_1*, *dataset_2* and *dataset_3*.
Our hypothesis is to find some keywords of the communities found and see how the nearby words around it change through the three word's network that we have built from the datasets.

A. Network building

The procedure, that we'll report now, used to build the network is the same for all three datasets (respectively $network\_1$,$network\_2$ and $network\_3$) therefore we'll report a simple analysis of the obtained networks only for one of these (the first in time order) because the results are very similar to each other. At first instance, it was necessary to carry out some operations of pre-processing on tweets, identified at the end of the scraping phase. Punctuation, useless words and characters, articles , words shorter than 3 letters and longer than 30 letters have been removed. It was not possible to carry out a more in-depth analysis, as most of the bookstores have been implemented only in English, and the implementation in Italian would have required more time.
However we used stopwords, i.e. the most frequent words within a language that do not add meaning or information to sentences, manually adding to them other stopwords, such as adverbs and conjunctions, used (rightly) a lot but without carrying any type of information. At this stage we make list of all the words (with repetition) present in every tweets then using **FreqDist** from the **NLTK**(Natural Language Toolkit) library we obtain a frequency distribution for the words and so use the its count as a ranking. Once we have built the ranking, we removed the words that have rank value smaller than a certain threshold (we use $min_{rank} = 4$).
Then we generate, for all the tweets, the bigrams between each words in it and save these in a list. In this list so are recorded the connection between words that appear in the same tweet at this point we select the 500000 most common bigrams, sorted according to their frequency (from the most common to the least common), and again using the **networkx** library, we create a network graph starting from the previously created dictionary of bigrams and their counts.
The result is a network with:

- *Number of nodes*: 13587
- *Number of edges*: 408172
- *Density*: 0.004

Using the fit function used in Sec:III, we obtain a power law distribution with a $\gamma = 2.01$, confirming once again that most real networks follow a power law, in particular a scale free regime being the coefficient between 2 and 3.

As can be seen in fig. 23 the starting point for the fit is 20, since the distribution for nodes with smaller degree no longer follows the exponential trend. This can be addressed to the cleaning process combined with the choice of selecting the first 50000 most common bigrams which lowered the number of words with few links(i.e. small degree).

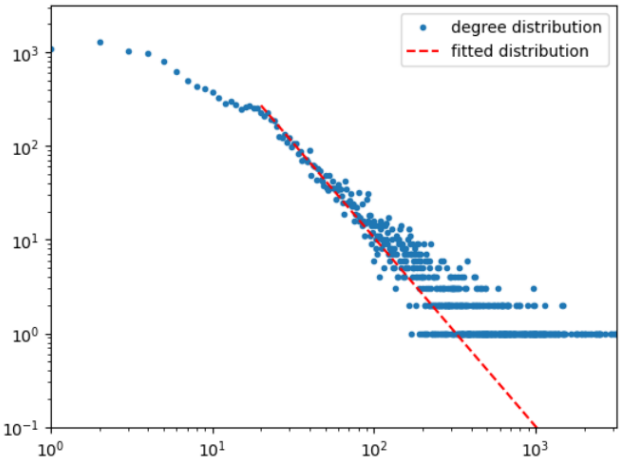

Fig. 23: Degree distribution $network\_1$

B. Community of words discovery

Looking at the results of Sec:IV we decided to use the **Leiden algorithm** for the CD on the obtained words network. Again we make an analysis of the min_size of the communities versus the $node\ coverage$ the results are reported in Fig:24.

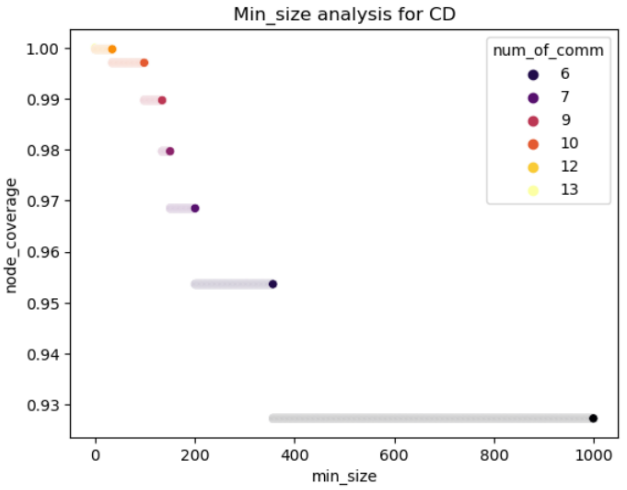

Fig. 24: Min_size analysis for CD on $network\_1$

At first glance it is noticeable that on the networks built in the previous section the number of communities found is significantly smaller then the number found in Subsec:IV-B, this due to the fact that we mentioned at the end of VII-A , furthermore it maintain a $> 90\%$ of node coverage for high min_size. In tab:VII we reported some stats of the CD analysis obtained. We can see how the value of $modularity$ has increased by an order of magnitude, as we could expect given the number of communities extremely reduced compared to before.

So we can explore more in dept the communities as tab:VIII where we reported the length and the average internal degree of all communities.

So excluding the last two communities (12, 13) in fig:25 we can see the communities graph. As in sec:IV-B we can see some closely connected communities, represented by those with more words (comm[0-4] in Tab:VIII), and then the less populated communities around.

	Leiden
# of comm.	14
max_int_degree	64
min_int_degree	1
modularity	0.16
hub dominance	0.50
conductance	0.66

TABLE VII: Internal stats Leiden on $network\_1$

	community length	avg_degree
$comm[0]$	5390	13
$comm[1]$	2264	68
$comm[2]$	1974	16
$comm[3]$	1521	18
$comm[4]$	1068	10
$comm[5]$	539	12
$comm[6]$	300	8
$comm[7]$	178	7
$comm[8]$	152	9
$comm[9]$	105	5
$comm[10]$	68	9
$comm[11]$	30	8
$comm[12]$	2	1
$comm[13]$	2	1

TABLE VIII: Length and avg_degree of the found communities

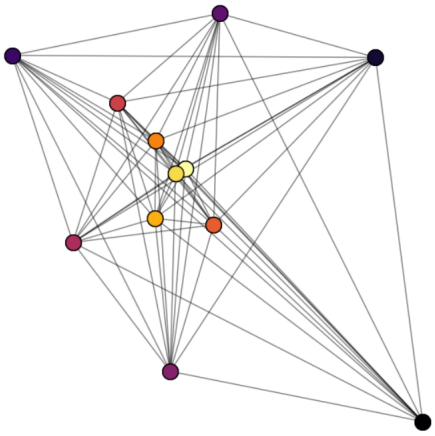

Fig. 25: Communities graph for $network\_1$

Fig. 26: WordCloud of the most significant communities of $network\_1$

C. Communities WordCloud

For better understand the CD made on the three network we will report the WordCloud , of the first 100 words(rank ordered), obtained for the first six community for each network that as we have seen before guarantee anyway an high *node coverage*, but trying to make everything as intelligible as possible we selected the two most significant for each group (fig:26-27-28).
Already at first glance we can notice variation in the "voter's dictionary". An example is that words related to governing semantics ("decree", "govern", "minister" , "premier"etc.) doesn't appear(or at least they appear with less importance) in the first network, while tend to gain greater importance in the last two.
Not less important is the fact that trough community detection we bring to light events that characterized the time interval of the correspondent dataset. In fact while in the first we found communities that concern the elections, already in the second dataset we succeed the detect events like *Qatar gate* or *Soumahoro case* (fig:27) and the in the third we detect the regional elections and an outlier concerning football match due to the fact that in February Europeans cups matches were played (fig:28).

D. Vocabulary changes around Keywords

At this point, it was decided to continue the analysis by choosing some keywords, like *bollette* (bills), *migrants*, *PNRR* (we report results only for the first), and calculate the wordclouds of the neighbors in the various datasets. The goal is to see if there has been any noticeable vocabulary change, as time has gone on. However, doing the wordcloud of the closest neighbors, we noticed that (as it should bee) the words mentioned were always the same, i.e. the names of the main politicians and their parties, as can also be seen from the figure 26. Therefore, we used a for loop that skips the n closest neighbors, so that we see the main words, avoiding the first ones - without additional information. Various attempts at n have been made manually, up to a range of 25-60, in which there seem to be some interesting vocabulary changes. In particular, using "bills" as a keyword, the following wordclouds were found (for the three respective datasets, in chronological order): Although most words, due to limited preprocessing, carry no information, several things can still be noticed. In the first dataset, relating to the election period, words such as *destra*, *sinistra*, *stato*, *politica* are mentioned, which indicates little interest in the problem of expensive bills and gas, in favor of the upcoming elections. In the second, once the new premier has been elected, there are passing words, such as *premier*, *rdc*, *soldi*, *migranti*. Finally in the last one, with the arrival of winter and the now more concrete problem of the price of gas, new words such as *benzina*, *caro*, *casa*, *Italia*, *milardi*, appear in the wordcloud.

VIII. Discussion and Conclusions

To sum up we can divide our report in two sections.

- The first comprehend a user-interaction network where the edges kept track of the tweet content.
- The second section considers a word-based network, extracted from the previous graph, where links determine the use of the two words in the same tweet with relative count.

Both sections represent a scale-free network with the gamma parameter ranging from 2-3, similar to most real social networks.

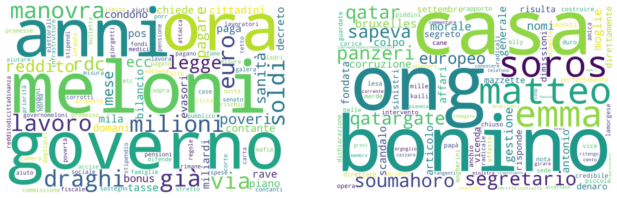

Fig. 27: WordCloud of the most significant communities of $network\_2$

Fig. 28: WordCloud of the most significant communities of $network\_3$

A. User Nerwork

We conducted a base topological and structural analysis of the first section.
The Leiden community algorithm was found to provide the best results in terms of node coverage, respect of the min-size of the clusters, modularity, and conductance score. The communities found by this algorithm represent the political clusterization of users well, as indicated by further analysis of the most common words in each cluster, as shown in Table IV and Table V. It is evident from these tables that each main cluster represents a different political party.
All basic opinion dynamic models have been used to show how a particular opinion would spread through the network. Additionally, communities of the leiden algorithm have been employed to capture the spatial ideological differences among the users (nodes). By using the CognitiveOpinionDynamic algorithm, we were able to successfully replicate the actual outcome of the elections. The crucial parameter for achieving this result was the value of ”external information = 0.9”. This value is consistent with the high volume of diverse information that can potentially reach a user on a social network like Twitter.
Finally, unsupervised link prediction algorithm have been used to anticipate possible interaction between the most linked nodes (i.e. with degree greater than 40). The comparison with the test set (network built with the interaction in the following months) are really poor. In fact the ROC curve has an area of ~ 0.01 when ideally should reach the unit. This problem can be addressed to the fact that the test network is built based on the tweet found with the keywords and not keeping track of users actions in the following months which was not possible with TwitterScraper.

Fig. 29: WordCloud for the word ”bollette”, using n=25.

B. Words Network

In this second section our goal was to, qualitative and quantitative, outline the variability of the political vocabulary as the time pass by taking into account 3 different word-graphs for 3 different spanning time.
Tweets have been preprocessed in order keep only the important words which could give an insight of the topic and opinion. Of course this was not an easy task as the Italian language is hard to interpret from single words and there are not many online packages ready to use for this purpose.
Once again the Leiden community algorithm have been exploited and a straightforward application of WordCloud already capture the topic diversity among different timespan.

- Dataset-1: The main topic are the different political parties and the imminent elections.
- Dataset-2: Right after the election the main words now are the actual premier and the opposition
- Dataset-3: At the time events takes over and topics like football league come up.

Furthermore we inspect the changes around particular words (in this case ”bill”).

- Dataset-1: Pre-election time, no particular events regarding bills occurred, main topic related are the political party.

- Dataset-2: After-election, ”bill” word is strongly connected to the RDC topic, one of the main subject at the time for the new premier.
- Dataset-3: Winter time, plus the well-known Ukraine war caused a raise in the fuel prices. Hence the main topic is now ”Benzina”.

Results are significant, although it is not possible to quantitative define the goodness of those achievements, the changes give an insight of the different topics characterizing those timespans.
This is not meant to be an exhaustive analysis, in fact further analyzes can be done, for example by implementing many of the NLP algorithms present on the internet in English and making them usable also in Italian.

IX. Bibliography

References

[1] Bright, J. et al. Does campaigning on social media make a difference? Evidence from candidate use of twitter during the 2015 and 2017 uk elections. Commun. Res. 47, 988–1009 (2020).
[2] P. Barberá, J. Jost, J. Nagler, J. Tucker, and R. Bonneau. Tweeting from left to right: Is online political commu- nication more than an echo chamber? Psychological science, 26, 08 2015.
[3] Radicioni, T., Pavan, E., Squartini, T., Saracco, F.: Analysing Twitter Semantic Networks: the case of 2018 Italian Elections. arXiv (2020).
[4] Giannetti, D. et al. Populism and Policy Issues: Examining Political Communication on Twitter in Italy 2018-20, Italian Political Science, VOLUME 15 ISSUE 2, SEPTEMBER 2020
[5] Liben-Nowell, David, and Jon Kleinberg. ”The link-prediction problem for social networks.” Journal of the American society for information science and technology 58.7 (2007): 1019-1031.
[6] Jingwei Wang, Yunlong Ma, Yun Yuan, Towards Fast Evaluation of Unsupervised Link Prediction by Random Sampling Unobserved Links.
[7] Coscia, M., Rossetti, G., Giannotti, F., Pedreschi, D. (2012, August). Demon: a local-first discovery method for overlapping communities. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 615-623). ACM.
[8] Benchmark graphs for testing community detection algorithms Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi Complex Systems Lagrange Laboratory (CNLL), Institute for Scientific Interchange (ISI), Viale S. Severo 65, 10133, Torino, Italy (Dated: October 30, 2008)