Graph clustering on Twitter: diagnosing polarization of the French public debate

Aurélien Stumpf-Mascles
MVA 2023-2024
aurelien.stumpf-mascles@polytechnique.edu

Meilame Tayebjee
MVA 2023-2024
meilame.tayebjee@hec.edu

Damien Vilcocq
MVA 2023-2024
damien.vilcocq@polytechnique.edu

***Abstract*—Community detection on social media networks has proven to be very efficient. We use a graph dataset containing thousands of interactions between Twitter users during the 2017 French presidential election to perform graph clustering using different algorithms and detect communities in an unsupervised fashion. We prove that the clustering yielded is 1) of high quality with regards to classic metrics 2) highly correlated with the political affiliations of the users, highlighting in a quantitative manner the polarization of the French political public debate.**

I. Introduction

A. Context

On the one hand, polarization of the political debate has been a highly discussed issue over the past few years, internationally. The social media, especially Twitter, have been accused of exacerbating this surging bigotry by recommending posts that match with your pre-existing opinion rather than opening the debate ([1], [2]).

On the other hand, Twitter, thanks to its network-like structure and the ease of collecting its data, has been central in several research areas, at the articulation between computer science (community detection) and social science (study of interactions). Relevant works about Twitter and political polarization detection include [3], [4].

B. Our contribution

This work is based on a graph dataset proposed by *Fraisier et al* in [5] [1] which gathers thousands of Twitter accounts, their corresponding tweets and retweets as well as political affiliations. We implemented several clustering algorithms found in reviews [6], [7], and [8] and partitioned the graph dataset into meaningful clusters (or communities). We first compared the clusterings constructed by each algorithm according to precise evaluation metrics (as discussed in [9]). We then interpreted these various clusters according to the political affiliations of the nodes using comparision metrics as introduced in [10] and highlighted the concept of echo chamber on social media.

II. Clustering

A. Clustering/Community detection problem

In graph theory and network analysis, the clustering or community detection problem refers to the task of identifying groups of nodes in a network that are more densely connected to each other than to the rest of the network. Mathematically, a graph $G = (V, E)$ consists of a set of vertices V and a set of edges E connecting the vertices. The goal of clustering is to partition the set of vertices V into disjoint subsets (clusters or communities) $C_1, C_2, \ldots, C_k$ that are fairly independent compartments of a graph.

However, there is no single universal definition of a "community" or "cluster" in a graph (group of nodes being more connected to each other than with the rest of the graph, group of nodes sharing similar structural positions, etc.), leading to various interpretations and methods based on different structural properties.

B. Metrics

The evaluation of clustering quality is not guided by a singular measure due to the multifaceted nature of clusters. Various metrics offer insights into different aspects of clustering quality, providing a comprehensive assessment.

1) Intrinsic Clustering Evaluation: Firstly, we consider metrics evaluating the clustering per se:

Intra-Cluster Density: This metric measures the average density within clusters, reflecting how closely related the members of each cluster are. Mathematically, it is expressed as:

$$\text{Intra-Cluster Density} = \frac{1}{|C|}\sum_{i=1}^{|C|}\frac{2|E_i|}{|V_i|(|V_i|-1)}$$

where C is the set of clusters, E_i is the set of edges within cluster i, and V_i is the set of vertices in cluster i. A higher value indicates tighter clustering.

Inter-Cluster Density: This complements the intra-cluster density by measuring the sparsity between clusters. It is defined as:

$$\text{Inter-Cluster Density} = \frac{1}{|C|}\sum_{i=1}^{|C|}\frac{|\text{Links}_{ext}(C)|}{|C|(|V|-|C|)}$$

where $|Links_{ext}(C)|$ is the number edges between a node of C and a node outside C. Lower values are desirable as they indicate well-separated clusters.

Modularity : Modularity quantifies the strength of division of a network into clusters. It compares the actual density of edges within communities to the expected density if edges were distributed at random, given the nodes' degrees. The definition

[1]The dataset can be found here https://zenodo.org/records/5535333

varies for directed and undirected graphs:
Undirected Graphs:

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

where: A_{ij} represents the edge weight between nodes i and j, k_i and k_j are the sum of the weights of the edges attached to nodes i and j, m is the sum of all of the edge weights in the graph, c_i and c_j are the communities of the nodes, δ is the Kronecker delta function, which is 1 if $c_i = c_j$ and 0 otherwise.
Directed Graphs:

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

where: k_i^{in} is the sum of the weights of the incoming edges to node i, k_j^{out} is the sum of the weights of the outgoing edges from node j.

2) Comparative Metrics: Next, we use metrics to compare the produced clusters C against the partition by political affiliation P:

Purity: This measure compares the dominant class of each cluster to the ground truth partition. The formula is:

$$\text{Purity}(P, C) = \sum_k \frac{\max_j |C_k \cap P_j|}{N}$$

where N is the total number of nodes, C_k is the set of nodes in cluster k of the obtained clustering C, and P_j is the set of nodes in the political class j of the partition P. Higher purity indicates a closer match to the true partition and hence shows the ability of the clustering algorithms to output communities mainly sharing the same political affiliation.

Normalized Mutual Information (NMI): NMI is an information-theoretic measure that assesses the shared information between the obtained clustering and the political partition:

$$\text{NMI}(P, C) = \frac{2 \times I(P, C)}{H(P) + H(C)}$$

where $I(P, C)$ is the mutual information between the clustering labels C and the political partition labels P, and $H(\cdot)$ denotes the entropy. $H(P)$ represents the entropy of the political partition. NMI values close to 1 imply a good match.

C. Algorithms

In this section, we present the different graph clustering algorithms that we implemented on the graph dataset. Characteristics of the algorithms are gathered in table I.

1) Hierarchical Clustering: The aim of community detection is to identify clusters based solely on the topology of the graph and eventually reveal its underlying structure. One of Twitter's trademark is the possibility to share other person's tweets, making it intuitively hierarchical with numerous small twitter accounts retweeting a few popular tweeter accounts. Therefore we decided to first implement an algorithm of hierarchical clustering, one of the most common clustering procedures.

TABLE I: Characteristics of community detection algorithms – W stands for edges weights and D for edges direction – n is the number of nodes and m is the number of edges

Algorithm	W	D	Complexity
Hierarchical Clustering	✓	×	$O(n^3)$
Louvain Clustering	✓	×	$O(n \log(n)))$
Spectral Clustering	✓	✓	$O(n^3)$
Label Propagation	✓	×	$O(m)$
Spectral Modularity Maximization	✓	×	$O(n(n+m))$

We implemented an agglomerative algorithm which iteratively merges the most "similar" clusters. The algorithm is given in 1. The similarity metric used is the sum of the weights from a cluster C_x to a cluster C_y.

Algorithm 1 Hierarchical Clustering

1: **Input:** Undirected graph G and number
2: **Output:** Clusters $C_1, C_2, \ldots, C_k$ (i.e., cluster assignments of each node of the graph)
3: **Initialization:** We start with the trivial clustering where each node is ina separate cluster (singletons)
4: **while** there is two distinct clusters
5: Compute similarity bewteen each cluster
6: Choose the pair of cluster (C_x, C_y) with highest similarity and merge them

The final cluster is then retrieved by choosing the best cluster according to the evaluation metrics defined before.

2) Louvain Algorithm: The modularity Q introduced in II-B1 is one of the most used quality metric to evaluate clustering. We implemented a greedy approach introduced by Blondel et al. in [11] for the general case of weighted graphs. The algorithm consists of iteratively constructing a locally maximal clustering for the modularity Q. The algorithm is based on the iteration of two phases (Modularity Optimization and Community Aggregation) until no improvement of the modularity is possible anymore. The first phase iterates over all vertices of the graph, displacing each vertex from its initial community to one of its neighbors when the move increases the modularity, and stops when no improvement has occured during a loop. The second phase then reconstructs a new graph where nodes are the communities found during the first phase and weights are the weights of the edges between each cluster. The algorithm is described in 2.

The main advantage of this algorithm is that it produces clusters of high quality with a very good time complexity of order $O(n \log n)$ where n is the number of nodes of the graph. This lies on the fact that it is sufficient to compute the difference of modularity ΔQ to make a move, where this quantity is very easy to compute and given by

$$\Delta Q = \frac{k_{i,in}}{m} - \frac{k_i^{out} \cdot \Sigma_{tot}^{in} + k_i^{in} \cdot \Sigma_{tot}^{out}}{m^2}$$

, where k_i^{out}, k_i^{in} are the outer and inner weighted degrees of node i and $\Sigma_{tot}^{out}, \Sigma_{tot}^{in}$ are the sum of in-going and out-going links from nodes in C.

Algorithm 2 Louvain Clustering

Input: Undirected graph G and number of clusters k
Output: Clusters $C_1, C_2, \ldots, C_k$ (i.e., cluster assignments of each node of the graph)
$G_{var} \leftarrow G$
while There is improvement
  for each node x **do**
    for each neighbour y of x in G_{var} **do**
      compute $\Delta Q = \frac{k_{i,in}}{m} - \frac{k_i^{out} \cdot \Sigma_{tot}^{in} + k_i^{in} \cdot \Sigma_{tot}^{out}}{m^2}$

3) Spectral Clustering: We implemented a spectral clustering algorithm which works on an undirected graph by leveraging the spectral properties of the Laplacian matrix of the graph. This method consists in solving a relaxation of the normalized graph cut problem (partition a graph in a manner that minimizes the cost of cutting between different clusters while maintaining clusters large enough). The intuition is that an approximation of a solution to this problem could provide quality clusters.

The Laplacian matrix plays a critical role to solve the relaxed problem as demonstrated in [12]. Its eigenvalues and eigenvectors provide insights into the connectivity and partitioning properties of the graph. Specifically, the second smallest eigenvector can be used to partition the graph in a way that roughly minimizes the cut between sets. By extending this concept to multiple eigenvectors, spectral clustering effectively finds a partition that balances minimizing the cut and keeping the clusters approximately equal in size.

The popularity of this algorithm comes from its simple implementation. With a theoretical complexity in $O(n^3)$, its use may be limited for very large graphs, although in practice the algorithm benefits from the fact that it uses only matrix manipulations, which are very well implemented. One drawback, is that the value of k must be selected. We selected the one that gave the partition maximizing the modularity of our graph.

Algorithm 3 Spectral Clustering

Input: Undirected graph G and number of clusters k
Output: Clusters $C_1, C_2, \ldots, C_k$ (i.e., cluster assignments of each node of the graph)
Let A be the adjacency matrix of the graph
Compute the Laplacian matrix $L_{sym} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. Matrix D corresponds to the diagonal degree matrix of graph G
Apply eigenvalue decomposition to the Laplacian matrix L_{sym} and compute the eigenvectors that correspond to d smallest eigenvalues. Let $U = [u_1 u_2 \ldots u_k] \in \mathbb{R}^{m \times k}$ be the matrix containing these eigenvectors as columns
For $i = 1, \ldots, m$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U. Apply k-means to the points $(y_i)_{i=1,\ldots,m}$ and cluster them in $A_1, A_2, \ldots, A_k$
Define the final clusters $C_1, C_2, \ldots, C_k$ such that $C_i = \{j | y_j \in A_i\}$

4) Label Propagation: We implemented a label propagation algorithm from [13]. This algorithm only uses the structural properties of the network and does not require neither the optimization of a predefined objective function nor the expected number of clusters.

It relies on the following procedure: each node is initialized with a unique label and, at each step, each node adopts the label that most of its neighbors currently have (is picked randomly if several communities are maximal and equally represented among the node's neighbors). During this iterative process, groups of densely connected nodes acquire the same label.

An asynchronous updating method is employed where node x at iteration t updates its label based on the majority label among its neighbors (that can have already been updated). The iterative process continues until a stable state is reached where every node has a label to which the maximum number of its neighbors belong to. Formally, it stops when after an iteration over all nodes, no change of label has been observed and for every node i, if i has label C_m then $d_{C_m} \geq d_{C_j}$ for all j, where d_{C_j} is the number of neighbors node i has with nodes of label C_j. This conditions ensures that no oscillations of labels can happen.
Finally, nodes that share the same label are grouped together as communities. The partition is composed of clusters that tend to be consensual (in the sense that a node belongs to the same cluster as the relative majority of its neighbors).

The strength of this algorithm lies in its simplicity (complexity in $O(m)$ and easy implementation). One drawback could be the randomness of the method, which means that the clusters produced are not always the same. To make the method more robust, [13] proposes a procedure to aggregate the results of several clusterings obtained by the label propagation method to obtain a more stable clustering. Given two different solutions, [13] define the aggregation as follows: let C_1 denote the labels on the nodes in solution 1 and C_2 denote the labels of the nodes in solution 2. Then, for a given node x, we define a new label as C_x = (C_x^1,C_x^2). Finally, we repeat iteratively this procedure for as many solutions as we want (usually robustness appears for a very low number of aggregated solutions). Our experiments showed that the aggregation of 5 solutions was enough to reach a stable clustering.

Algorithm 4 Label Propagation

For each node x, let $C_x(t)$ be the community associated to node x at iteration t
Define f, the function that at iteration t given a node x, will output the majority community among x neighbors. If several communities are maximal and equally represented, the new community of x doesn't change if x was already in one of these community or otherwise is picked randomly
Initialize each node x with a unique label $C_x(0) = x$
$t \leftarrow 1$
while Labels continue to change **do**
    $order \leftarrow$ random permutation of all nodes in the network
    for each node x in $order$ **do**
        $C_x(t) = f(x)$
    end for
    $t \leftarrow t + 1$
end while
return Final cluster of all nodes

5) Spectral Modularity Maximization: Spectral Modularity Maximization is a spectral method designed for undirected graphs that follows an approach analogous to the Louvain Algorithm, aiming to output a clustering with good properties by attempting to maximize modularity. This method is grounded in the principle that the eigenvectors of the modularity matrix can be used to partition the network into communities, which inherently optimizes the modularity score.

The spectral method employs the eigenvectors of the modularity matrix to assign nodes to communities. Indeed, when one considers the split of a graph G in two groups, the modularity of the split graph can be written:

$$Q = \frac{1}{4m} s^T B s$$

where B is called the modularity matrix and verifies $B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$ and $s_i = 1$ if node i belongs to group 1 and $s_i = -1$ if it belongs to group 2.

Finding the split that maximizes the modularity corresponds to the following optimization problem:

$$\max \frac{1}{4m} s^T B s \text{ subject to } s_i \in \mathbb{Z}, s_i^2 = 1 \forall i$$

This integer programming problem is intractable and the idea exposed in [14] is to use the following relaxation:

$$\max \frac{1}{4m} s^T B s \text{ subject to } s \in \mathbb{R}^n, \sum_{i=1}^{n} s_i^2 = n$$

Using Lagrange optimality conditions, one can obtain that a solution of the relaxation is an eigenvector of B. Then, the maximal modularity that can be obtained is $\frac{n\lambda}{4m}$ where λ is the eigenvalue associated to s. Hence, the maximization of the modularity is obtained taking s^* the eigenvector associated to the largest eigenvalue of B.
s^* is not a solution for our first problem since s_i^* may not belong to $\{-1, 1\}$. The choice is to use s such that $s_i = \text{sign}(s_i^*)$ as an approximation of the optimal solution.

As we have seen, we are able to find a good division of a graph in two clusters. This approach can be generalized to handle the division of graphs in more than two clusters. The classical approach proposed in [14] is repeated division into two: using the spectral bipartition presented above to cut the graph in two parts, then divide those parts, and so forth. More precisely, the idea is to authorize only the divisions that results in an increase of the graph modularity and rely on the easy computation of the change in modularity, as in the Louvain's algorithm.

Algorithm 5 Spectral Modularity Maximization

Initialize a queue Split to contain the set of all nodes (i.e., the partition in one cluster). Split will contain the clusters that may potentially be split.
Initialize an empty queue Partition that will contain the final partition.
while Split is not empty **do**
    Subset $\leftarrow$ Dequeue(Split)
    Compute S_1 and S_2, the spectral bipartition of Subset
    Compute ΔQ, the difference in modularity of the graph induced by the bipartition of Subset
    if $\Delta Q > 0$ **then**
        Enqueue(Split, S_1)
        Enqueue(Split, S_2)
    else
        Enqueue(Partition, Subset)
    end if
end while
return Partition

III. Methodology

A. The dataset

This work is based on a graph dataset proposed by *Fraisier et al* in [5] [2]. It gathers 22,853 Twitter accounts, their corresponding tweets and retweets, plus the retweet and mention networks related to these profiles. The team also provided their political affiliations (up to two political parties among LFI, PS, LREM, LR and FN), their nature (individual or collective), and the sex of the profile's owner. Edges of the graph therefore connects user i to user j if the former has retweeted tweets from the latter. The weight of the edge is equal to the number of total retweets from one user to the other.

B. Preprocessing

As the original graph is heavy, we had to apply preprocessing to the dataset to decrease its size while preserving its structure. The preprocessing steps are the following :

1) We sampled a given proportion of the nodes (70% for example) while keeping the same political distribution in the graph
2) We only kept the nodes which had an outer weight of more than 3, which means that we only keep tweeter accounts which have retweeted more than 3 times the same account.

[2]The dataset can be found here https://zenodo.org/records/5535333

We observed that clustering results were much better when we did the second preprocessing part. This could mean that most active tweeter accounts are also the most politically engaged whereas less active tweeter accounts could be connected to various political opinions.

Finally, we observed that Spectral Clustering, Spectral Modularity Maximization and Label Propagation algorithms had better results when we made the graph undirected with strictly positive weights equal to 1. Making the graph undirected increases the number of edges which generates more stable results. Putting the weights to 1 also makes the inversion step of the Spectral Clustering algorithm more stable.

IV. Results

A. Code

All the code experiments are contained in the *experiments.ipynb* notebook. Two packages are also provided containing auxiliary functions :

- *clustering* package contains the different clustering algorithms as well as the metrics
- *utils* package contains the functions used to plot

B. Results

TABLE II: Comparison of clustering algorithms based on different metrics

Clustering Algorithm	Intra	Inter	Mod	T
Hierarchical Clustering	6.5e-01	5.1e-04	1.2e-01	3.3e+01
Louvain Algorithm	7.0e-01	1.1e-04	1.6e-01	9.9e-01
Spectral Clustering	6.6e-01	6.8e-05	1.1e-01	1.7e+00
Spectral Modularity Max	6.5e-01	6.9e-05	1.0e-01	3.0e+00
Label Propagation	6.6e-01	1.4e-04	1.1e-01	1.1e+00

Table II summarizes the results of our clustering algorithms. All in all, all of our clusterings achieve great performance with regards to the metrics defined in II-B1. All of these algorithms have been trained unsupervisingly (agnostically to the political affiliations of the users) ; some of them do not require a target number of clusters (Label Propagation, for instance), and those which need one have been provided with $k = 5$, the number of political parties. Overall, all of them achieve similar results, Spectral Clustering and Spectral Modularity Maximization having a slight edge in terms of Inter-Cluster Density.

We can conclude saying that these clusters are of high quality. Let's see how they relate to the political affiliations: Table III summarizes the results.

TABLE III: Comparison of clustering algorithms based on the metrics NMI and Purity with regards to political affiliations

Clustering Algorithm	NMI	Purity
Hierarchical Clustering	6.9e-01	9.3e-01
Louvain Algorithm	8.5e-01	9.6e-01
Spectral Clustering	8.2e-01	9.4e-01
Spectral Modularity Maximization	8.7e-01	9.6e-01
Label Propagation	8.6e-01	9.6e-01

Once again, all of our algorithms achieve similar performance. With $\approx 95\%$ Purity on average compared to the political affiliations, and a NMI $\approx 85\%$, the link is clear. The algorithms yield clusters that are very close to the political partition. Something even more interesting is the fact that an algorithm such as Label Propagation - which do not require a target number of clusters - exactly converges to five clusters, each cluster being a political affiliation. We discuss that more deeply in IV-C.

Basically, we conclude that the clusters are indeed of high quality intrinsically and happen to correlate almost exactly with the political affiliations. This happens for *all* of our algorithms, which shows how the partition following the political affiliations is strong, clear and natural.

C. Discussion

The computed clusters clearly demonstrate the phenomenon of echo chamber on Twitter as each cluster is extremly polarized towards one political party.

For instance, the Label Propagation algorithm (II-C4), which does not need a target number of clusters, yields by itself five clusters - exactly the number of political parties. To dive deeper, we plot in A, for each of the five clusters, the histogram of the political affiliations. We can see that the clusters obtained match almost perfectly with the political affiliations. The analysis can be extended to the other clustering algorithms, see Appendix (A).

In other words, we say that political affiliations are both *natural* and *strong* clusters : clustering algorithms converge to this repartition even when trained agnostically to the affiliations, and yield high quality clusters.

V. Conclusion

As discussed in IV-C, all of our unsupervised algorithms yield clusters containing users only from one party. In IV-B, we showed that these clusters were *good* with regard to certain metrics (II-B1).

We conclude that Twitter users, *at least those who are involved in the political debate*, tend to interact only with people that think like them. The clusters of political affiliations in the Twitter space are strong, dense and clearly separated ; a situation that highlight either the fact that Twitter plays a role in the polarization of the public debate, or that the political debate has become polarized and that Twitter interactions only reflect that reality.

References

[1] S. Hong and S. H. Kim, "Political polarization on twitter: Implications for the use of social media in digital governments," *Government Information Quarterly*, vol. 33, no. 4, pp. 777–782, 2016.

[2] E. Kubin and C. von Sikorski, "The role of (social) media in political polarization: a systematic review," *Annals of the International Communication Association*, vol. 45, no. 3, pp. 188–206, 2021.

[3] A. Makazhanov and D. Rafiei, "Predicting political preference of twitter users." New York, NY, USA: Association for Computing Machinery, 2013. [Online]. Available: https://doi.org/10.1145/2492517.2492527

[4] I. Weber, V. R. K. Garimella, and A. Batayneh, "Secular vs. islamist polarization in egypt on twitter." New York, NY, USA: Association for Computing Machinery, 2013. [Online]. Available: https://doi.org/10.1145/2492517.2492557

[5] "Élysée2017fr: The 2017 french presidential campaign on twitter," vol. 12. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14984
[6] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, 2007.
[7] Z. X. S. D. e. a. Ding, Z., "Overlapping community detection based on network decomposition. sci rep 6, 24115 (2016)."
[8] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0370157309002841
[9] U. Brandes, M. Gaertler, and D. Wagner, "Experiments on graph clustering algorithms," vol. 2832, 11 2003.
[10] O. Fraisier, G. Cabanac, Y. Pitarch, R. Besançon, and M. Boughanem, "Uncovering like-minded political communities on twitter." New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3121050.3121091
[11] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, oct 2008. [Online]. Available: https://dx.doi.org/10.1088/1742-5468/2008/10/P10008
[12] U. von Luxburg, "A tutorial on spectral clustering," 2007.
[13] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, Sep. 2007. [Online]. Available: http://dx.doi.org/10.1103/PhysRevE.76.036106
[14] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, p. 8577–8582, Jun. 2006. [Online]. Available: http://dx.doi.org/10.1073/pnas.0601602103

APPENDIX

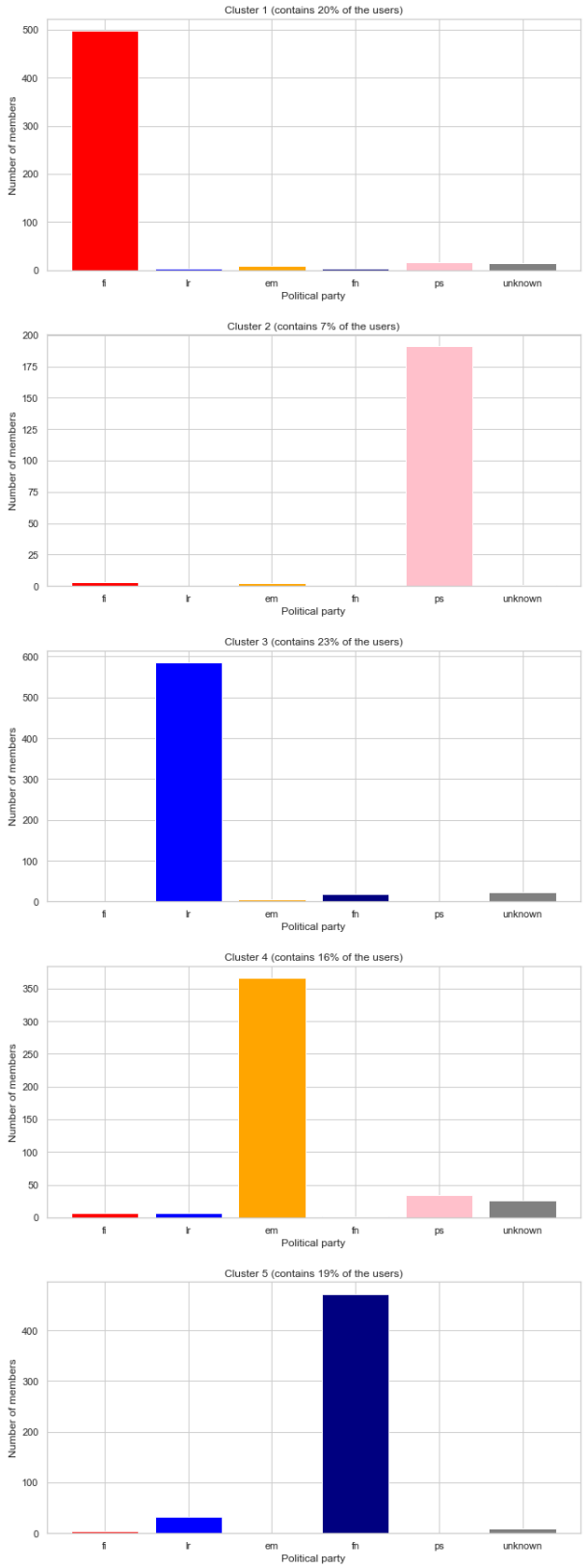

Fig. 1: Repartition of the political affiliations within the significative (containing more than 1% of the nodes) clusters obtained using Hierarchical Clustering

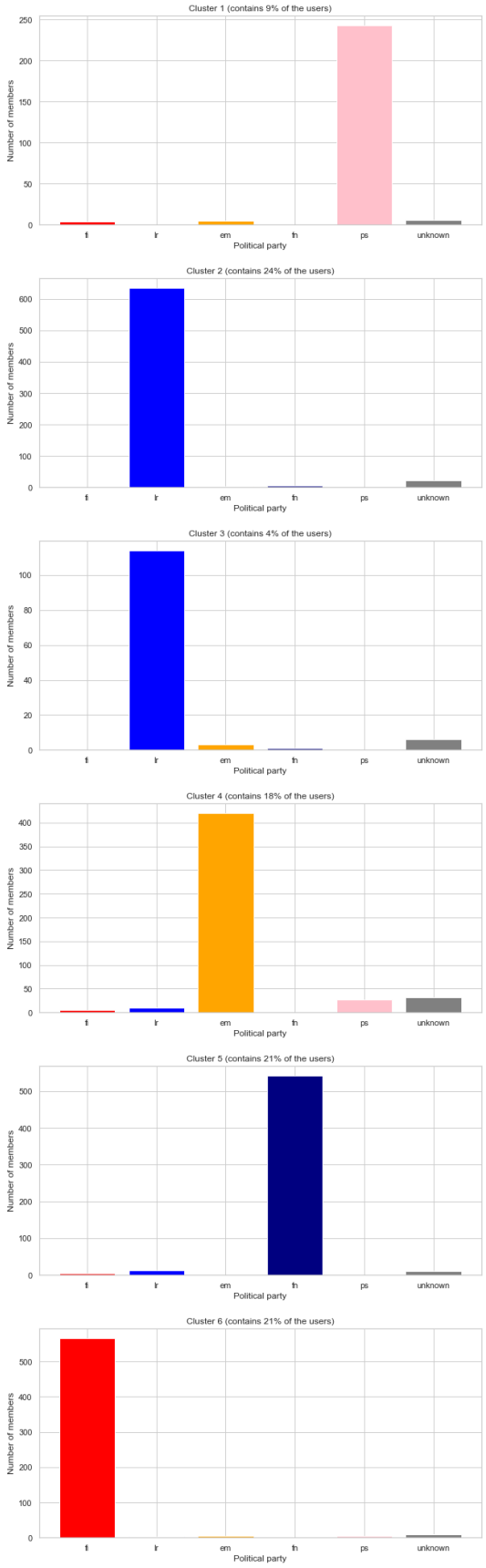

Fig. 2: Repartition of the political affiliations within the significative (containing more than 1% of the nodes) clusters obtained using Louvain Algorithm

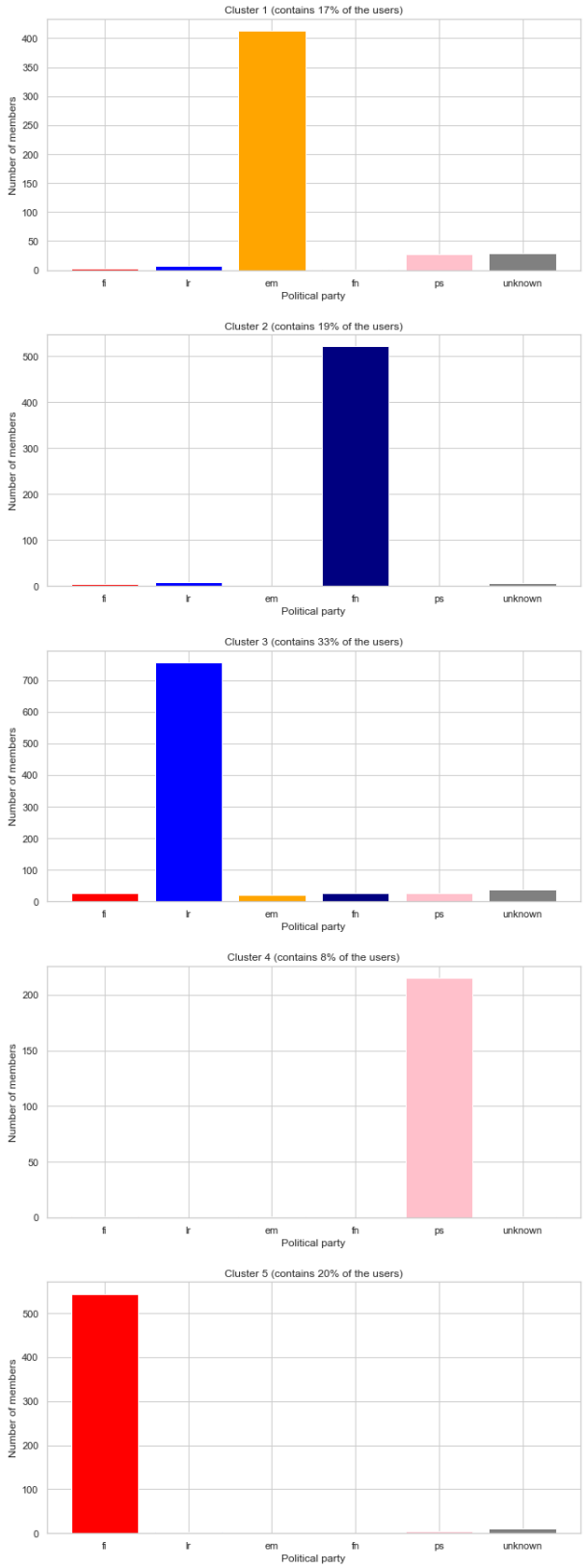

Fig. 3: Repartition of the political affiliations within the significative (containing more than 1% of the nodes) clusters obtained using Spectral Clustering

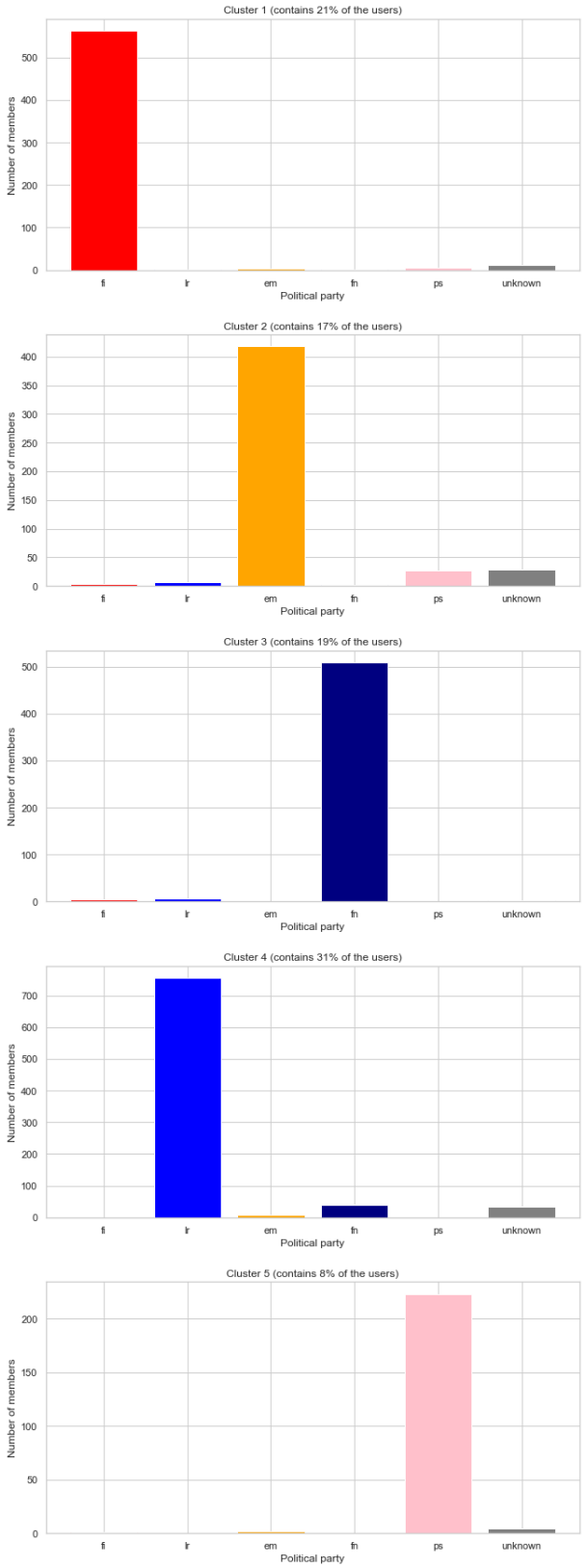

Fig. 4: Repartition of the political affiliations within the significative (containing more than 1% of the nodes) clusters obtained using Label Propagation Algorithm

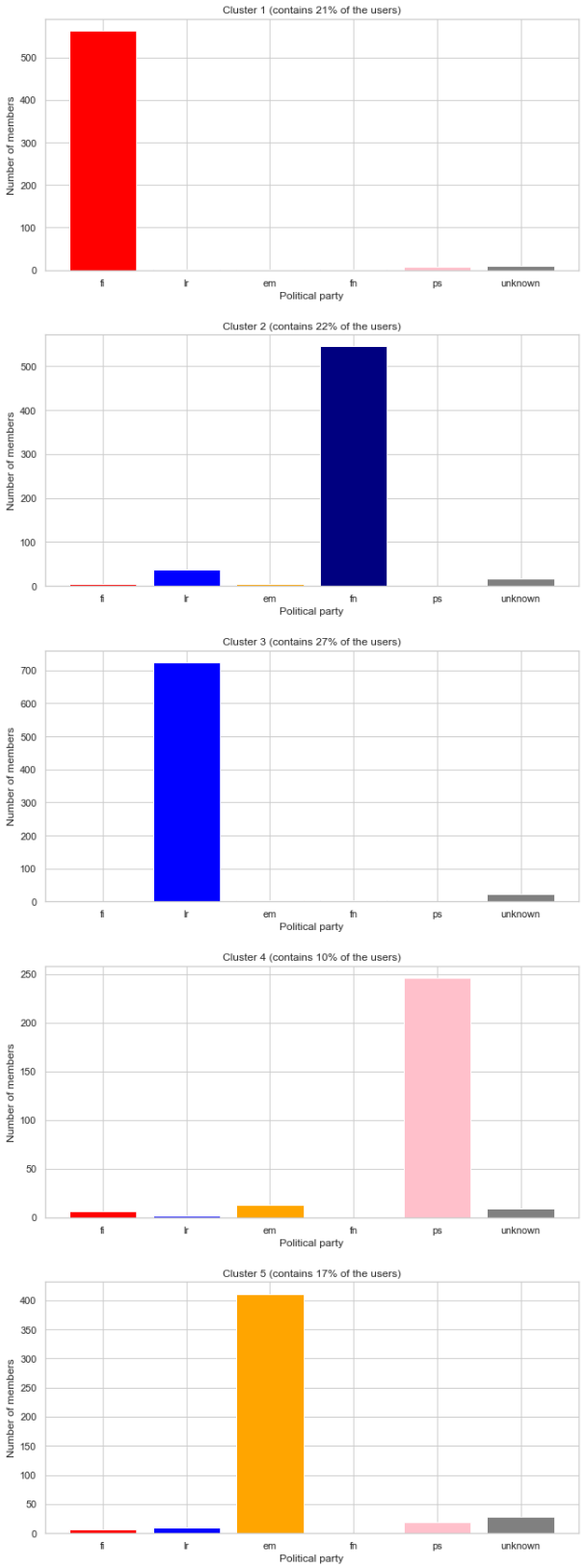

Fig. 5: Repartition of the political affiliations within the significative (containing more than 1% of the nodes) clusters obtained using Spectral Modularity Maximization Algorithm