Statistical Analysis of Network Data
Course by Eric D. Kolaczyk
Project paper: Comparison of community detection methods in Facebook social networks

Alexandre Combessie* David Duong-Prunier† Ismail Machraoui‡

December 31, 2015

Abstract

This project paper intends to explore the characteristics of a sample of Facebook ego-networks and to compare the performance of several unsupervised community detection methods on this dataset. We implement 4 methods: Leading Eigenvector, Walktrap, Markov Cluster (MCL); and Hierarchical Clustering, which accounts for mixed community membership. We use Facebook profiles to compute a similarity measure between users and weigh the network edges accordingly. We compute the Balanced Error Rate to evaluate the performance of each method with respect to the real social circles. The MCL method outperforms the others although it does not allow for mixed membership. We also find that similarity weights significantly improve the performance of the Walktrap and Hierarchical Clustering methods.

1 Introduction

For this project paper, we chose to apply our new knowledge of Statistical Analysis of Network Data [1, 2] to a common topic of interest: social media. In particular, we decided to study Facebook, the most prominent social networking website with 1.55 billion monthly active users as of September 30, 2015 [3]. We leveraged the vast literature on the subject, in particular [4, 5], and anonymized ego-network datasets made available on Kaggle [6] by Jure Leskovec and Julian McAuley.

*Mainly in charge of data loading, descriptive analysis, and similarity weighting. Email: alexandre.combessie@ensae.fr

†Mainly in charge of researching and implementing community detection methods. Email: david.duong.prunier@ensae.fr

‡Mainly in charge of data loading, feature preparation and performance evaluation. Email: ismail.machraoui@ensae.fr

Besides understanding the key characteristics of these ego-networks, our goal is to automatically identify the various social circles among one's Facebook friends, in an unsupervised way. Since we have the real social circles in our dataset, as labeled by the egos, we can compare our prediction with the truth, and evaluate the performance of each method. The topic of community detection is important for all social networks, and especially for Facebook, which does not require users to label their social circles, contrary to Google+. From our own personal experience as Facebook users, it can be difficult to manually sort out a long list of friends. Automatically detecting communities can help users save time, and improve the overall Facebook experience through a better curation of the news feed.

Our paper is organized as follows. In section 2, we examine the data available and undertake some exploratory network characterization in order to have an overall idea of our ego-networks. In particular, we study our networks' order, size, degree distribution, centrality measures, and cohesion. We also analyze the Facebook profile data available on each user, and social circle labels given by the ego. Afterwards in section 3, we will describe the selected community detection methods from the associated literature, and illustrate it with our ego-networks. We will also describe how we can turn Facebook profile information into vertex attributes and weigh the network edges based on attributes similarity. In section 4 we will introduce our performance measure, the Balanced Error Rate, and explain how to compute it. Thus, we will run a performance comparison between our community detection methods, with or without similarity weights.

2 Descriptive analysis of our dataset

The dataset [6] comprises of 110 ego-networks extracted from the online social network Facebook. Vertices are defined as users, i.e. the ego and his or her friends. Edges represent mutual friendship, with no information on the "nature" and "intensity" of the friendship relation, so the network is undirected and unweighted. Some user profile information is also available for each vertex, such as the user's name, hometown, schools, and employers (past and present). All data has been anonymized, and there is no way to relate each network with each other.

Besides, for all ego-networks, we have labels given by the ego user for the friends in their social circles. However, not all users have labels, as the ego users have provided partial information. Finally, in order to evaluate the performance of our community detection models, we have been provided the ground-truth circles, or might we say the "real" human-labeled circles, supplied by the ego users, for 60 ego-networks (not all of the 110).

2.1 Visualization

In the next figure, we have plotted a sample of 4 ego-networks. We see that the order (number of vertices) and size (number of edges) of the network varies, with sparse and dense networks. As emphasized by the Fruchterman-Reingold layout, communities tend to appear visually.

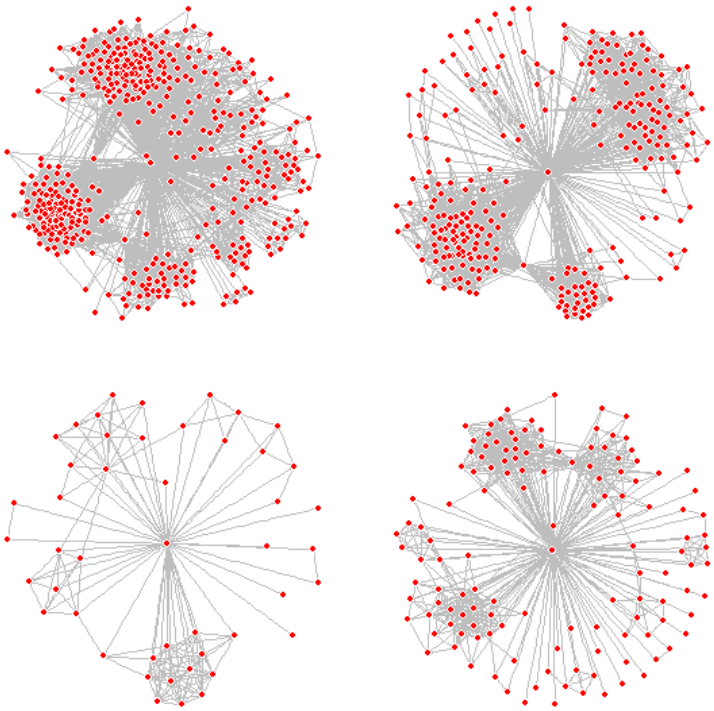

Figure 1: Visualizations of a sample of networks, with the ego included

Note that we have included the ego for visualization purposes, but from now in this section, we will exclude the ego from our analysis, to facilitate the interpretation.

2.2 Network order and size

To better understand the characteristics of these ego-networks, we studied the distribution of the order and size of these networks (ego excluded). Here, order is simply the number of friends of the ego, and can be interpreted as a measure of sociability. Size is the number of friendship relations between friends of the ego. It is hard to interpret it in terms of network cohesion as studied in subsection 2.5.

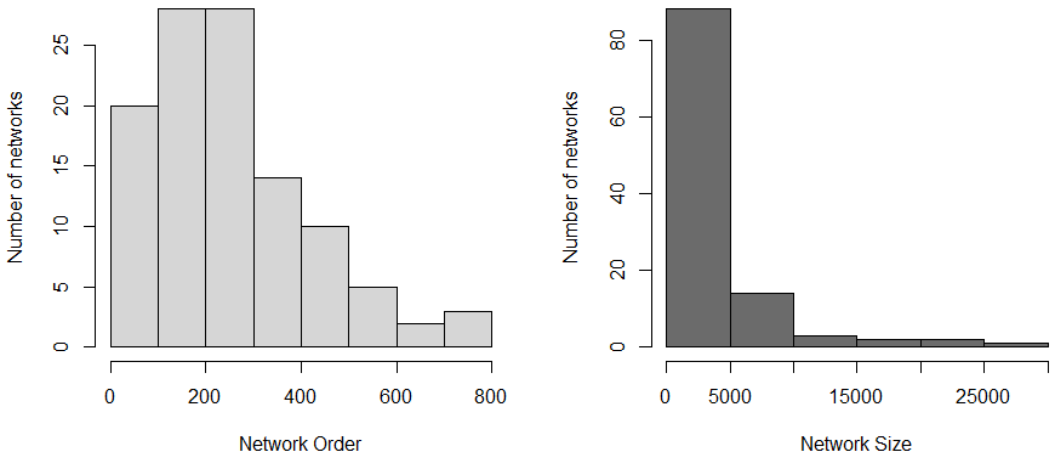

Figure 2: Distribution of the order and size of all 110 networks, without the ego

	Order	Size
Minimum	12	14
1st Quartile	119	670
Median	215	1686
Mean	251	3388
3rd Quartile	338	4129
Maximum	781	25290

Table 1: Summary statistics on the order and size of networks

On average in our sample, an ego has 251 friends. The sample of 'egos' considered is relatively diverse, from a seemingly 'solitary ego' with 12 friends to a 'socially active' one with 781 friends. The order distribution is relatively smooth (Skewness of 1.10, Gini of 0.37), with a visually linear decay.

As for network size, the distribution is more skewed and unequal (Skewness of 2.74, Gini of 0.60), with a visually exponential decay.

2.3 Degree distribution

To better characterize our sample of networks, we will study its degree distribution[1], i.e. the number of mutual friends with the ego for a given user. It can be interpreted in terms of "sociability" or "importance" of each user *within* the network induced by friendship with the ego. Since each of our networks is only a sampled subgraph of the complete Facebook graph, it is a poor measure of real sociability or importance. To take an example, a popular user could have no mutual friends with the ego but still have a large group of friends not captured in the dataset. Therefore, we will be careful not to over-interpret this measure.

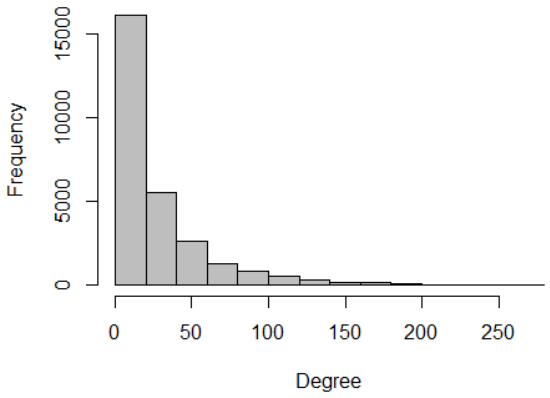

Figure 3: Degree distribution for all networks combined

	Degree
Minimum	0
1st Quartile	6
Median	16
Mean	27
3rd Quartile	35
Maximum	276

Table 2: Summary statistics on the degree distribution

On average for all networks combined, a user has 27 mutual friends with the ego. We see that the degree distribution is skewed and unequal (Skewness of 2.34, Gini of 0.57). As studied in many empirical works on social and information networks, our degree distribution seems to follow a power law [7, 8, 9]. To verify it, we can apply a very simple Ordinary Least Squares model[2].

[1]Since the networks are unweighted, degree is equal to strength.

[2]It is only a first naive step, as more robust approaches exist today, such as maximum-likelihood fitting with goodness-of-fit tests based on the Kolmogorov Smirnov statistic and likelihood ratio [7].

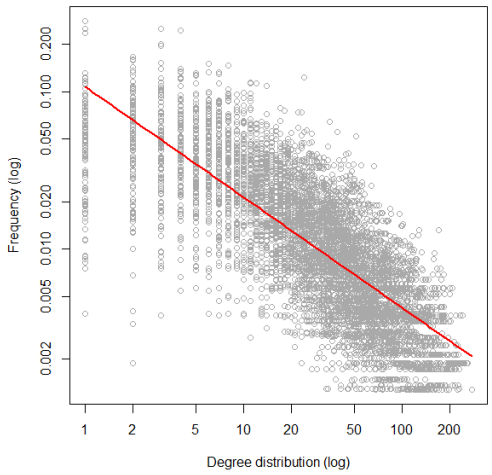

Figure 4: Degree distribution for all networks combined, fitted to a power law

	Dependent variable:
	log(frequency)
log(degree)	-0.701^{***}
	(0.008)
Constant	-2.238^{***}
	(0.027)
Observations	6,034
R^2	0.586
Adjusted R^2	0.585
Res. Std. Error	0.660 (df = 6032)
F Statistic	$8{,}522.404^{***}$ (df = 1; 6032)
Note:	*p<0.1; **p<0.05; ***p<0.01

Table 3: Summary of the OLS regression statistics

We see that the OLS regression coefficients are statistically significant at 99%, provided the OLS normal assumptions are verified. However, to test the correct specification, we would need more in-depth models.

2.4 Network centrality

This is close to the previous section 2.3 on degree distribution. However, we will refrain from studying centrality in terms of flow of information in the network, because the ego-networks only offer a very incomplete view of the overall Facebook graph. In our dataset, centrality measures can only be interpreted as the relative social importance of the user strictly within the group of friends of the ego.

For all networks combined, we can plot the distribution of several classic centrality measures: Closeness, Betweenness, and Eigenvector. The distribution of these measures being very skewed and unequal (see table below), we have plotted the distributions in log scale. We note that Eigenvector centrality produces a less skewed distribution.

	Skewness	Gini
Closeness Centrality	16.72	0.63
Betweenness Centrality	27.57	0.86
Eigenvector Centrality	1.61	0.37

Table 4: Skewness and Gini index of centrality distributions

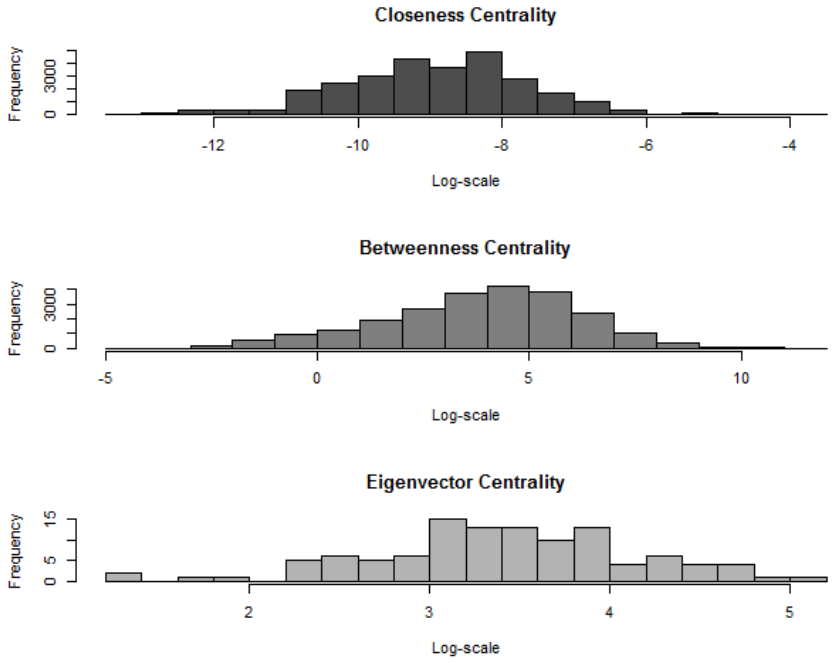

Figure 5: Centrality distributions for all networks combined

The skewed nature of these distributions could be interpreted as the fact that people have only a few "important friends" within a larger group of "acquaintances", who are less central in the ego-network.

2.5 Network cohesion

To further study the subject mentioned above, it is interesting to study our networks in terms of cohesion: how dense are the networks? Are there a lot of "solitary nodes" with no mutual friends? How many vertices are in the largest cliques?

To answer these questions, we will study several indicators:

- Density of each network, measured by $\frac{|E|}{|V|(|V|-1)/2}$
- Ratio of the giant component order on the total order, for each network: $\frac{|E_{giant}|}{|E_{total}|}$
- Distribution of order of components, for all networks combined
- Order of the largest cliques of each network

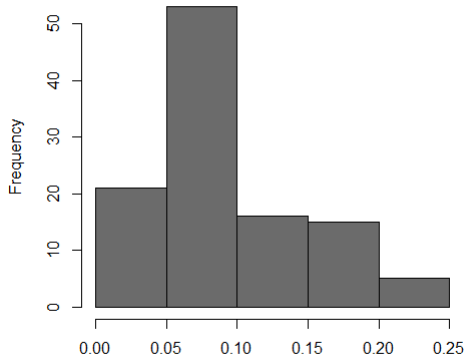

Figure 6: Distribution of the density of each network

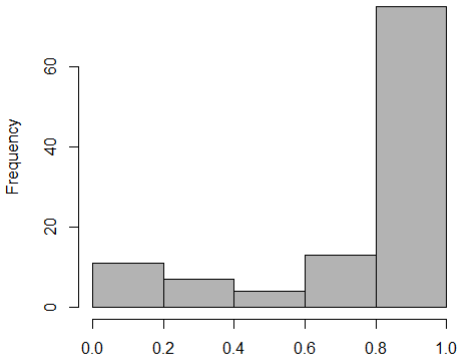

Figure 7: Distribution of the ratio of order of the giant component

We see that the networks are not very dense (mean density of 9.3%), and that the giant component represents the majority of the network (average of 76.5% of the total order of each network). Thus the large majority of friends of the ego are part of a subgraph of mutual friends.

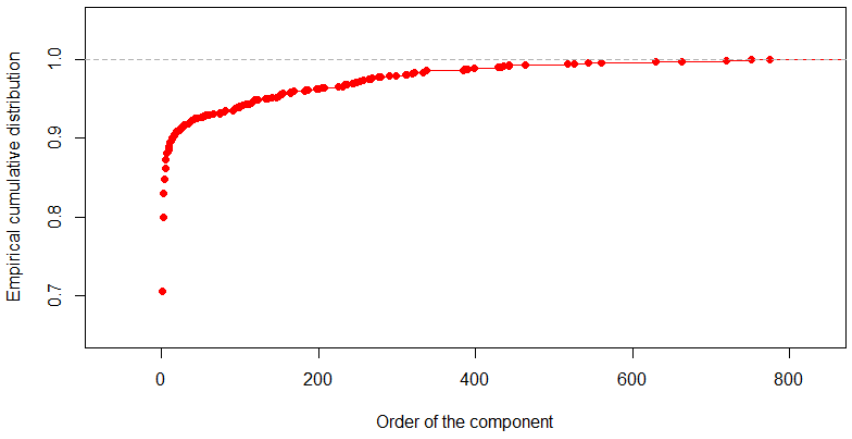

Figure 8: Empirical cumulative distribution of components' order for all networks

As shown in the figure above, the distribution is skewed: atomic components represent 70.5% of the count of components, but only 3.5% of the total order of all networks.

Finally, we will study the distribution of order of the largest cliques.

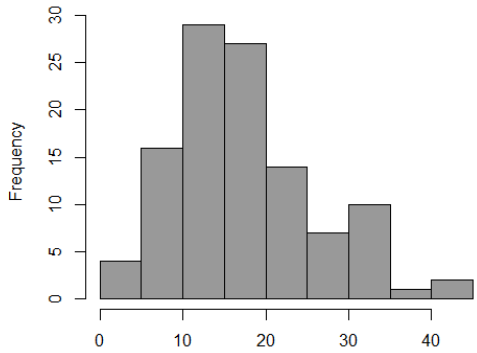

Figure 9: Distribution of order of the largest cliques in each network

	Order
Minimum	4.0
1st Quartile	12.0
Median	16.5
Mean	18.3
3rd Quartile	23.0
Maximum	42.0

Table 5: Summary statistics on the largest cliques' orders

We see that the largest clique encompasses on average 18 friends, which can be interpreted as the size of the group of "close friends" of the ego. However, the interpretation of "close friends" in this case is not the same as in section 2.4 on centrality.

2.6 Feature completion

In addition to the ego-networks, we have features from the Facebook profiles of each ego and friends. We flattened this feature data, which was originally under a tree format, to build a set of 56 vertex attributes[3]. However, not all attributes were filled by the users: the ratio of completion of all features for a given network is low. There is more than 50% of incomplete data for each network.

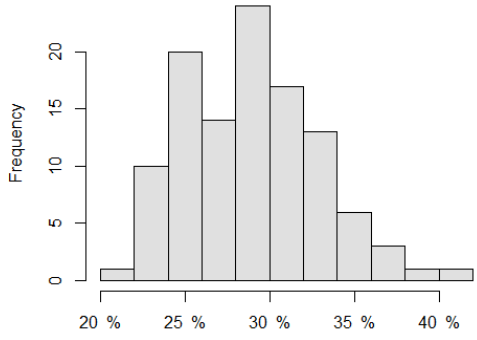

Figure 10: Distribution of the ratio of completion of all features by network

	Ratio
Minimum	20.9%
1st Quartile	25.6%
Median	29.0%
Mean	28.9%
3rd Quartile	31.3%
Maximum	41.2%

Table 6: Summary statistics on the ratio of completion

[3]When considering only "name" and not "id" types of features, the number is reduced to 36.

But the ratio of completion depends a lot on the features considered. Almost all users give their locale, name, gender and birthday information ; whereas very little information is available on the users' religion, political views, and work projects.

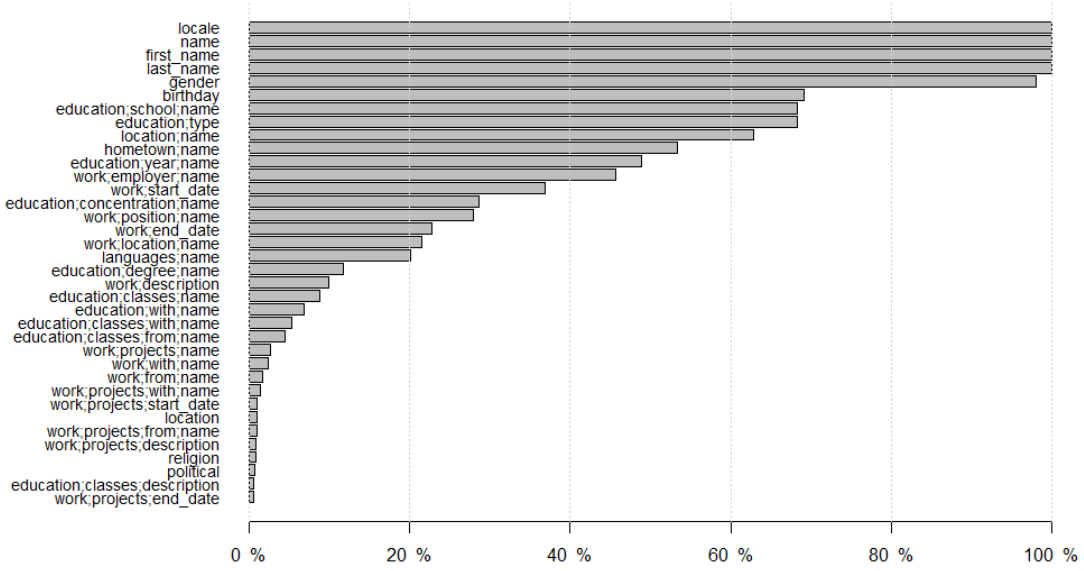

Figure 11: Percentage of completion by feature, for all networks combined

2.7 Analyzing user-labeled social circles

Based on social circle training data we have on 60 ego-networks, we can answer several questions: How many circles to consider? How many friends does a circle contain? Does everyone belong to a circle? Is there a lot of overlap between circles?

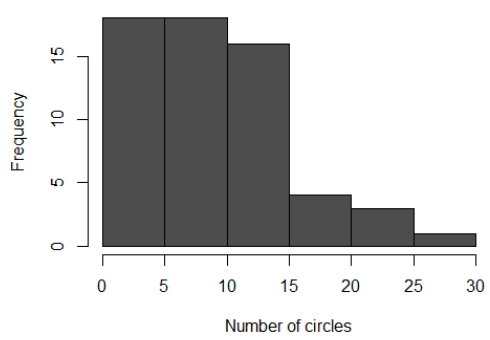

Figure 12: Distribution of the number of circles per ego

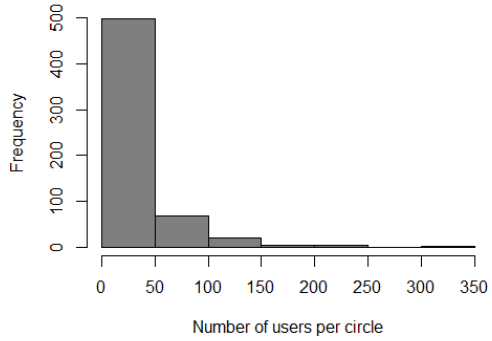

Figure 13: Distribution of the number of users per circle for all networks combined

From the previous figures, we see that the number of circles per ego is relatively uniform. It is less so for the distribution of number of users per circle. On average, each ego has defined 10 social circles, with 29 users per circle.

We also see that our training data is well labeled: on average in each ego-network, 80% of users have been labeled to a social circle. Finally, the phenomenon of mixed membership is very important: 37% of users are part of 2 or more social circles. We will need to take this into account in section 3 on Community Detection.

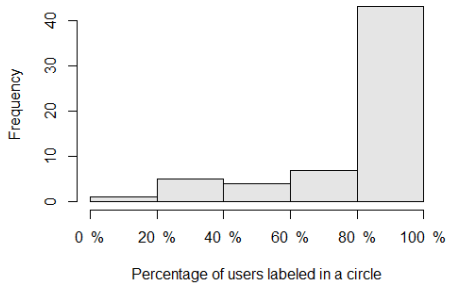

Figure 14: Distribution of the number of circles per ego

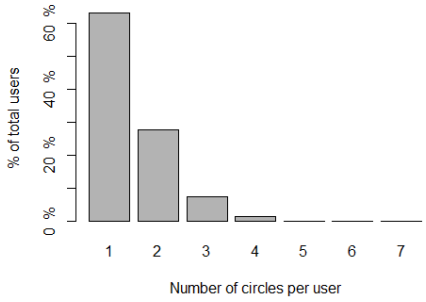

Figure 15: Distribution of the number of users per circle for all networks combined

3 Community detection methods implemented

Firstly, the notion of community needs to be precised here. While the literature defines it as group of people living in the same place or sharing common interest, our paper will mainly focus on connection degrees between individuals. This "network thinking" bias is not really far from reality as the intuition leads to think that the more people share common interests, the more likely they will have shared connections (A knowing B knowing C knowing A for example). The first part will be focusing on analysing the network and detecting community with the assumption that communities are formed solely by connections between alters, i.e regardless of the features. We will perform 4 algorithms: leading eigenvector, walktrap, MCL (Markov Clustering Algorithm) and hierarchical clustering. While the second part will take into account the features of each alters by putting a weight on each edge, the weight representing a features-based similarity measure.

3.1 Leading Eigenvector

The methodology developed by Mark Newman [10] is based on the analysis of the edges densities of the vertices that compose the network. It assumes that groups have strong communities structure with dense connection between all individuals of a same group. Mathematically, the problem is equivalent to maximizing the benefit function, also called the modularity function. While very effective in its original format, the algorithm has high computational requirement. Mark Newman propose an approach which consist in rewriting the modularity matrix B by using the adjacency matrix A such that:

$$B_{ij} = A_{ij} - P_{ij}$$

where A_{ij} the adjacency matrix of the network
and P_{ij} is the probability of a connection between two vertices i and j

Then calculate the eigen vectors u, take the most positive one u_{max}, and look at the sign of the components of u_{max} which should be a mix of positive and negative number forming two groups that form two communities. Formerly, given s an index vector with $\mid s \mid = n$, number of vertex, the modularity is maximal for s parallel to u_{max} eigen vector of B and we have :

$$s_i = \begin{cases} +1 & \text{if corresponding element in } u_{max} \geq 0 \\ -1 & \text{if corresponding element in } u_{max} < 0 \end{cases}$$

Zhang propose the following algorithm [11]:

Given modularity matrix Q

1. Compute the leading eigenvector of B.
2. Compute the index vector s and the modularity matrix Q
3. Repeat until Q is stable, subdivise into group otherwise

Community detection rendering on a sample network:

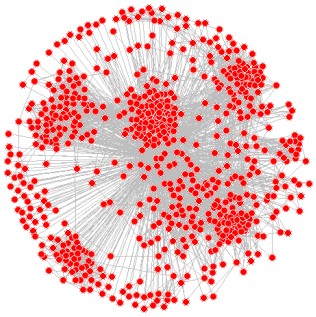

Figure 16: Original network without any grouping

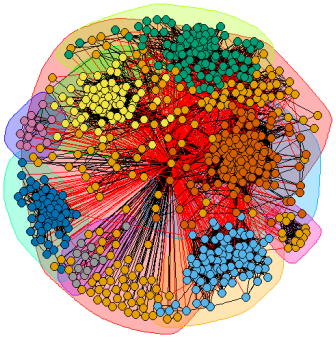

Figure 17: Leading eigen vector - 7 groups detected

	Values
Vertices	670
Edges	8556
Communities	7
True communities	21

Table 7: Summary of grouping results

The implementation of the algorithm is done by using the iGraph *leading eigen vector* class which allows us to run a quick, yet precise, test on our sample. The outcome shows us that the grouping is working but the results are too restrained by the number of iteration that it runs, grouping far more communities than the ground truth. Here the true groups are 3 folds the predicted communities. One hypothesis, and as a first step to enhance the precision of the algorithm, is that the communities are abnormally high in this network. (the average is 10, see descriptive statistics part of this paper). Indeed with 21 real groups, we suspect that some communities in the ground truth are in fact nested communities that are not taken into account in the eigen vector methodology. We will confirm that later by looking at the BER which shows far more better results overall.

3.2 Walktrap

The walktrap algorithm has been developed by Matthieu Latapy [12] and refers the intuition that a random walk in a graph will tend to be trapped in one circle (thus the name walktrap). This methodology, similarly to the previous one assumes that edges between two vertices of a same group are more probable than edges between two vertices from two different communities. Intuitively the edges are "concentrated" within circles. Latapy calculates the distance between vertices and applies a hierarchical clustering using Ward's agglomeration method to form communities.
Algorithm:

Start with partition $P_1 = \{\{v\}, v \in V\}$
Start with n single vertex communities
Compute distance between all adjacent vertices
iterate

1. Choose two communities C_1 and C_2, calculate distance r
2. Merge $C_3 = C_1 \cup C_2$ and create new partition $P_k = (P_{k+1} \ / \ \{C_1, C_2\}) \cup C_3$
3. Update existing distance between communities

Community detection rendering on a sample network:

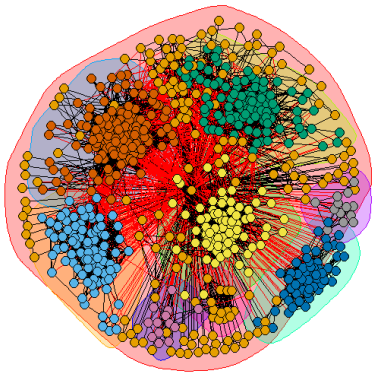

Figure 18: Walktrap algorithm applied to *Figure 16*

	Values
Vertices	670
Edges	8556
Communities	30
True communities	21

Table 8: Walktrap grouping results

The first remark is that the distance used in the algorithm seems rather suitable for community detection. Compared to leading eigen vector we have more granularity in the number of community detected.

3.3 MCL

The Markov cluster algorithm has been developed by Dongen (2000) [13]. The methodology is to simulate a random walk using Markov matrices and applying two operators alternatively namely the inflation and the expansion operators. While the

former is similar to taking the power of the Markov matrix, the latter uses the scaled Hadamard power of the matrix to obtain a new stochastic matrix. After n iterations the matrix eventually reach a state where it doesn't change neither with expansion nor inflation steps. The algorithm induce using knowing the network, that is using the adjacency matrix as an input:

Given G a graph
Set M1 to be the matrix of random walks on G
while (change)
1. M2 = M1 * M1
2. M1 = γ * M2
3. change = difference(M1, M2)

Community detection rendering on a sample network:

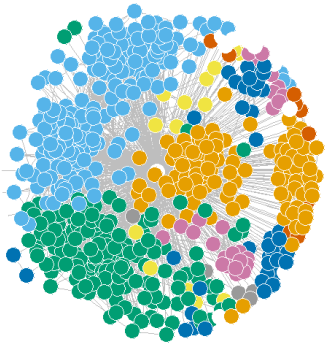

Figure 19: MCL algorithm applied to network *Figure 16*

	Values
Vertices	670
Edges	8556
Communities	2 5
True communities	21

Table 9: Hierarchical clustering grouping results

3.4 Hierarchical clustering

In the following, we use a single-linkage clustering [14] which is part of the family of hierarchical clustering method which involves calculating a distance D between each vertices. The idea behind hierarchical clustering is to maximize the inner distance and minimize the outer distance at each step. The single linkage clustering assumes that each vertex is a group of its own at step 1 and then sequentially agglomerate groups in which the distance between two vertices of each group is the minimum.
Algorithm:

Start with n groups composed of single vertex
Repeat until obtaining one group
1. Calculate all distances of all vertices' alters
2. Merge groups where distance D between two vertices from different group is minimum

Visually, it is easier to look at the grouping process through a dendrogram.

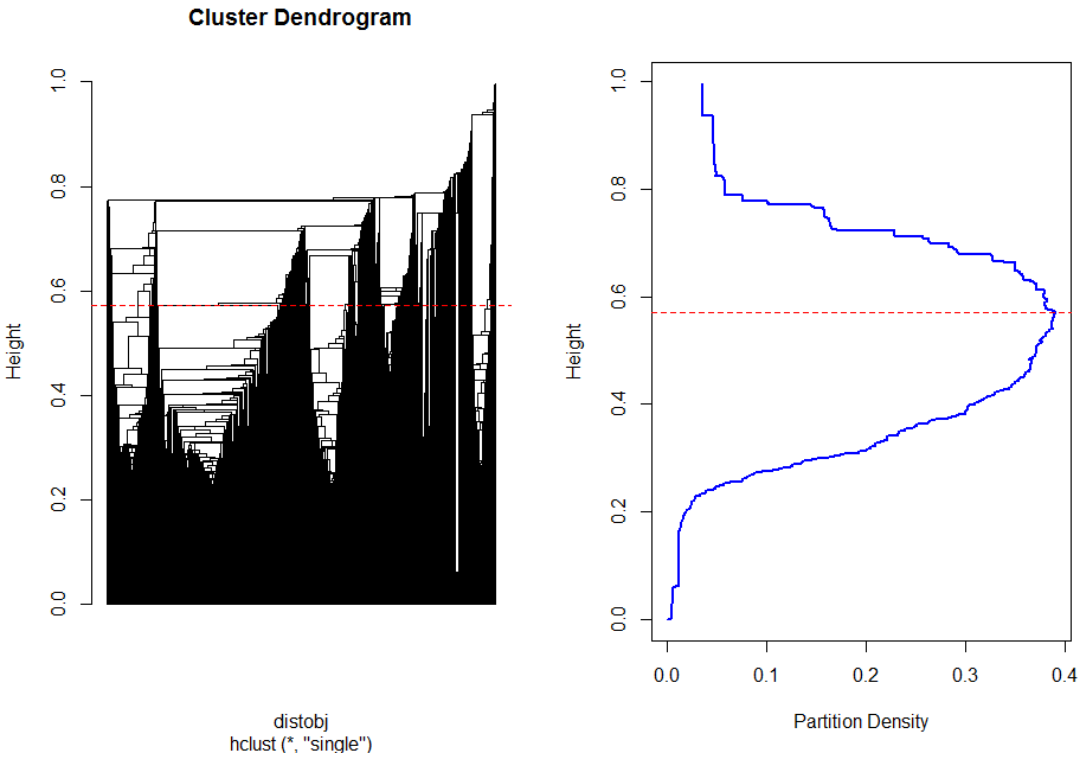

Figure 20: dendrogram of single linkage clustering with communities density

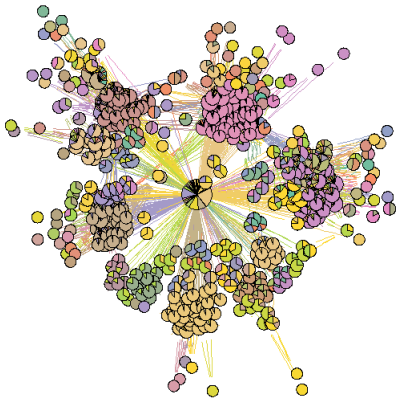

Figure 21: Hierarchical clustering algorithm on network *Figure 16*

	Values
Vertices	670
Edges	8556
Communities	242
True communities	21

Table 10: Hierarchical clustering grouping results

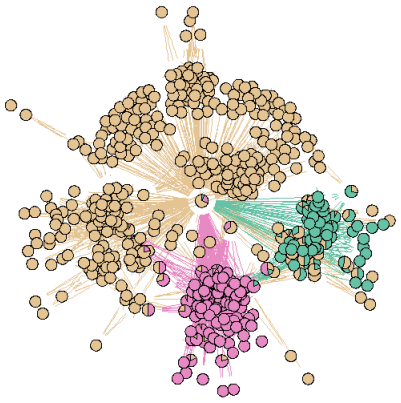

Figure 22: Merging nested communities

	Values
Vertices	670
Edges	8556
Communities	4
True communities	21

Table 11: Merging results

The R package we are using, namely Linkcomm, ease the application of the clustering algorithm to our sample network. One advantage of using the library is that it provides a good illustration of possible nested communities. In the example we applied two hierarchical clustering methodologies from the package. As we can see the first one has a very high cutting criteria (*fig. 21*), the number of community is very high and really not representative of reality. The second grouping takes the highest partition density (*fig. 20 fig. 22*) as cutting criteria. We obtain 4 groups (*tab. 11*). If look in detail the composition of the groups (*fig. 23*), we can see that the algorithm merges the group by taking into account nested communities, i.e. one vertex belonging to two communities. While this is going to cause large discrepancies in number of communities (4 vs 21), it is more accurate when closing the results to the ground truth data (see BER results in part 4)

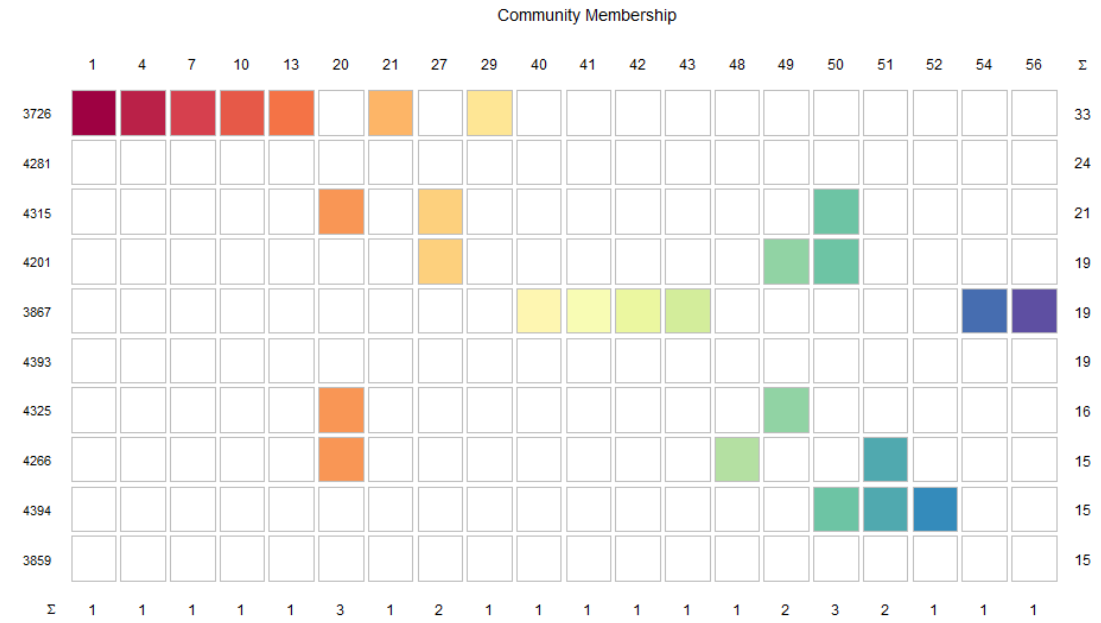

Figure 23: Nested communities using Linkcomm hierarchical clustering function
colored cells shows all nodes groups, bottom x-axis is the final group given by the algorithm

3.5 Using profile similarity to compute edge weights

Facebook profile attribute	Vertex 1	Vertex 2	Similarity
id	1	146	FALSE
last_name	1	140	FALSE
first_name	1	122	FALSE
birthday	1	64	FALSE
name	1	146	FALSE
gender	1	1	TRUE
locale	0	0	TRUE
hometown;id	1	67	FALSE
education;school;id	0,2,3,3	0,86	TRUE
education;type	0,1,1,1	0,1	TRUE
education;year;id	0,2,3	0,10	TRUE
education;concentration;id	1,2	36,70,71	FALSE
location;id	1	0	FALSE
education;classes;from;id	0		FALSE
education;classes;with;id	0		FALSE
education;classes;id	0		FALSE
work;position;id		82	FALSE
work;location;id		33	FALSE
work;start_date		2	FALSE
work;end_date		5	FALSE
work;employer;id		132	FALSE
languages;id		2,46	FALSE
work;with;id			FALSE
work;description			FALSE
work;from;id			FALSE

Table 12: Illustration of our profile attribute similarity measure for a given edge

As illustrated in the table[4] above, we implemented a very straightforward algorithm to compute our profile attributes similarity and add edge weights to our networks. Our similarity measure for two connected vertices v_1, v_2 is defined as:

$$\text{Similarity}(v_1, v_2) = \sum_{i=1}^{N_{attr}} \mathbf{1}\big\{\text{attr}_i(v_1) \cap \text{attr}_i(v_2) \neq \emptyset\big\}$$

With attr_i being the list of past and current values of the i-th attribute. It means that if any past or current value of the i-th attribute of v_1 is the same as that of v_2, we add 1 to our similarity measure. For instance, one user could be working at the same company as one of his friends who left the company. Note that we do not count any missing value.

Based on this similarity measure, we initialize all weights at 1, and loop over all networks and all edges to add our similarity measure to the weights.

[4]To simplify the table, we have only represented here a sample of all attributes. All values are represented by integers since our dataset was anonymized. Blank values mean that data is missing.

4 Conclusion: Performance evaluation

4.1 Balanced Error Rate

In order to compare the implemented methods previously stated, the chosen evaluation metric would be the Balanced Error Rate (BER).
The BER definition in a network-related problem would be as follows :

$$BER(C_P, C_T) = \frac{1}{2}(\frac{|C_P \setminus C_T|}{|C_P|} + \frac{|C_P^c \setminus C_T^c|}{|C_P^c|})$$

with:

- C_P : stands for computed circles
- C_T : stands for ground truth circles

In any given ego-network, we would end up with a (N_p x N_t) BER matrix, as for N_p being the number of computed circles and N_t being the number of true circles.

Furthermore, as the number of predicted circles are usually not equal to the ground-truth ones, we will have to deal with two cases :

- $N_p \leq N_t$: the assignment is quite straight forward. For any given predicted circle, the matching true circle would be the one with the maximum (1- BER) rate in any computed line of the BER matrix
- $N_p > N_t$: this case leads us to test all merging combinations of computed circles and hence calculate the BER matrix once again. The best combination possible would be the one maximizing the following formula:

$$\frac{1}{N_t} \sum_{i,j=1}^{N_t} 1 - BER(C_{mi}, C_{tj})$$

knowing that C_{mi} is a computed circle after a merging combination.

4.2 Results

Here below a synthetic table summing up all results of the community detection methods:

	1 - BER	
Model	**Unweighted**	**Weighted**
Leading Eigen Vector	68.9%	69.1%
Walktrap	57.5%	66.2%
MCL	70.7%	70.1%
Hierarchical Clustering	45.3%	60.9%

Table 13: Comparative performance of the methods implemented

Evaluation methodology: Due to the high computational cost of weighting and BER estimation[5], we have performed this evaluation on a subset of 20 ego-networks, and kept only the giant component. For the Leading Eigenvector and Walktrap methods, we also tuned the parameter controlling for the number of steps in order to avoid predicting too many circles compared to the reality (see subsection 2.7).

Comparing all percentages, forgetting at first about the weighted aspect, the MCL algorithm is the one showing the best result (70.7%). Even though weights are taken into account, the MCL would still be the winner. We notice that the added weights to the ego-networks are not significant enough to either the Leading eigen vector or the MCL, i.e the BER rate is quite similar for the unweighted and the weighted networks.

However, for the Walktrap and Hierarchical Clustering, with lower performance, the addition of weights significantly improves the BER. Thus, similarities between the Facebook profiles of two friends can be useful information to detect communities in combination with network data. It can also help interpreting each social circle in terms of shared attributes e.g. employer, school, etc.

[5]In the case where the number of predicted circles is higher than the number of true circles, the complexity becomes exponential as we need to test all merging combinations.

References

[1] E.D. Kolaczyk. *Statistical Analysis of Network Data: Methods and Models*. Springer Series in Statistics. Springer, 2009.

[2] E.D. Kolaczyk and G. Csárdi. *Statistical Analysis of Network Data with R*. Use R! Springer New York, 2014.

[3] Facebook company information page, 2015.

[4] Jure Leskovec and Julian J McAuley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.

[5] Julian McAuley and Jure Leskovec. Discovering social circles in ego networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(1):4, 2014.

[6] Julian McAuley. Learning Social Circles in Networks | Kaggle, 2015.

[7] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

[8] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[9] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, 2011.

[10] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E, vol. 74, Issue 3, id. 036104 (PhRvE Homepage)*, 2006.

[11] Yan Zhang. Community detection methods using eigenvectors of matrices. *http://netwiki.amath.unc.edu/uploads/Publications/Ma11Paper$_f$inal.pdf*.

[12] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *https://www-complexnetworks.lip6.fr/ latapy/Publis/communities.pdf*.

[13] Stijn van Dongen. Introduction to mcl. *http://www.micans.org/mcl/intro.html*.

[14] www. Wikipedia single linkage clustering. *https://en.wikipedia.org/wiki/Single-linkage$_c$lustering*.

[15] Lars Backstrom, Paolo Boldi, Marco Rosa, Johan Ugander, and Sebastiano Vigna. Four degrees of separation. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 33–42. ACM, 2012.

[16] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74:036104, Sep 2006.

[17] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks. *CoRR*, abs/cs/0702048, 2007.

[18] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005*, pages 284–293. Springer, 2005.

[19] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[20] Alex T Kalinka and Pavel Tomancak. linkcomm: an r package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics*, 2011.

[21] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.