LOUVAIN
School of Management

UNIVERSITE CATHOLIQUE DE LOUVAIN

LOUVAIN SCHOOL OF MANAGEMENT

COMPARISON OF SOME COMMUNITY DETECTION METHODS FOR SOCIAL NETWORK ANALYSIS

Promoteur : Marco Saerens

Mémoire-recherche présenté par

Augustin Collette

en vue de l'obtention du titre de
Master en sciences de gestion (en ingénieur de gestion)

ANNEE ACADEMIQUE 2014-2015

I would like to thank Pr. Marco Saerens for his presence and the help and support he provided during more than one year.
Also, thank you to Bertrand Lebichot and Nicolas Forêt for answering my questions.

Contents

1. Introduction

Nowadays, more and more data come in the form of networks especially in the growing world of Internet. Common examples are the World Wide Web with its pages and hyperlinks and social networks like Facebook and LinkedIn. Beyond Internet, the same is true for examples like citation networks of scientific literature or biological networks (e.g. protein interactions) [59]. Not only network data are more and more widespread, but they also have shown their incredible utility such as with the example of Google. Google leveraged the network structure of the web in order to rank the immense amount of pages by importance [41] and provide a powerful way to find information.

In this master thesis, we focus on trying to group the objects of these networks (i.e. people in the case of Facebook) into what we call clusters or communities (i.e. groups), based on closeness or similarity between these objects. Imagine a network of people where these people are not linked based on friendship but somehow linked because they have similar behaviours. Marketers would be very interested to identify in this network communities of people that share a certain type of behaviour, in order to target them. This is an example of business situation where graph clustering can be very valuable. This business value, in addition to the research potential of the field, is one of the main reasons why I chose to study the present topic.

In pattern recognition and data mining[1], clustering is considered as part of what is called "unsupervised classification methods" because clustering consists in categorizing data based on their similarity and difference, without using class labels if available beforehand. Clustering has applications in many fields such as social sciences, engineering and life sciences [52].

Furthermore, this thesis focuses on the use of kernels that provide a way to represent similarity between objects (that can be persons) in networks. Technically, kernels consist in the main input on which clustering algorithms work here. We will compare the quality of different kernels, each one being computed on a certain way to represent distance between objects. Indeed, there are many ways to measure distance between objects of a network.

A main issue in clustering is that often the number of clusters needs to be specified as input because it is not determined by the algorithm. In this thesis, we analyse the L method whose

[1] The two terms are equivalent

objective is to find the optimal number of clusters based on a set of clustering partitions for different number of clusters possible. We compare its effectiveness compared to the Louvain method, which is a clustering algorithm where the optimal number of clusters is determined by the algorithm.

Sometimes, there are different numbers of clusters that would make sense in an application and there is no "right answer". In those cases, the best thing is to give the outcome to an expert who will decide which one should be kept [52]. However, here, we take only one "natural" number clusters or classes as determined either by a human or by the context and we will judge the effectiveness of the L and Louvain methods according to this benchmark. One may argue that this research has no value because there is no definition of a "good" number of clusters and each method has its own way to define what is "good". Nevertheless, we argue that it is valuable to assess how "good" is the number of clusters found by those methods (in our case the L method and the Louvain method) compared to the best benchmark that we have at hand (i.e. the "natural" number of clusters).

The thesis is structured around the following research questions:

1. Which kernels give the best results when used in a Ward's hierarchical clustering algorithm? More specifically, ...

    1.1 Which of the following kernels gives the best clustering partition: the sigmoid commute time kernel or the one based on the free energy distance?

    1.2 Does the sigmoid transformation of the corrected commute time kernel gives better results than the same transformation applied on the commute time kernel?

    1.3 If the free energy kernel matrix is made positive semi definite (thus a mathematically valid kernel), does it give better results in a Ward's hierarchical clustering than the original free energy kernel?

    1.4 Do the randomized shortest path and logarithmic forest kernels produce better partitions than the previously analysed kernels?

2. Is the Ward's hierarchical clustering algorithm better than the Louvain method?

    2.1 Is the number of clusters found by the L Method used on a Ward's hierarchical clustering hierarchy more closer to the "natural" number of classes than the number found by the Louvain method?

    2.2 Does the Ward's clustering produce better partitions than the Louvain method?

The structure of this paper starts with an introduction of the theoretical concepts of graph, distance, kernel and clustering. Meanwhile, we cover the specific types of distance and kernel used in this thesis, as well as the two clustering algorithms of interest, namely the Ward's hierarchical clustering (and the associated L method) and the Louvain method. Then, in the experiments, after introducing the datasets and our experimental procedure, we present and discuss the results for each research question.

Part I.

Theory

2. Preliminaries

2.1. Notations

Some useful notations to know are:

- Column vectors are represented by bold lower case letters like $\mathbf{u}$ and matrices by bold upper case letters like $\mathbf{A}$
- $\mathbf{I}$ is the identity matrix, containing zeros everywhere except on its diagonal that is full of ones
- (m, n) represents the size of a matrix containing m rows and n columns
- $\mathbf{e}$ is the column vector full of ones of the appropriate size and $\mathbf{e_i}$is the i-th column of $\mathbf{I}$, containing zeros every where except on row i
- $\triangle_{ij}$ refers to the distance between 2 objects i and j, $\triangle^{(2)}$ is a square distance, and $\mathbf{\Delta}$ ($\mathbf{\Delta}^{(2)}$) is the corresponding matrix regrouping all (squared) distances
- $x_{i\bullet}$ and $x_{\bullet j}$ mean the sum of the i-th row and the sum of the j-th column of the corresponding matrix $\mathbf{X}$, respectively
- $\mathbf{diag}(\mathbf{X})$ is a column vector containing the diagonal of a square matrix $\mathbf{X}$
- $trace(\mathbf{X})$ is the trace of a square matrix $\mathbf{X}$, which is the sum of its diagonal elements

2.2. Networks or graphs

The foundation of this thesis is the concept of network or graph that we wish to introduce along with some definitions. The interested reader is encouraged to look at [39] for a more detailed introduction of networks.

Typically, in data analysis, it is assumed that all data instances are independent from each other. Nevertheless, often in reality those instances are linked through various types of relationships; sometimes in addition to be described themselves by various attributes [59]. This where networks come in, they provide a way to represent relationships between data objects.

A graph, or network, $G = (V, E)$ is a structure containing a set V of n vertices hereafter called nodes (i.e. the data instances or objects) and a set E of edges consisting in pairs of vertices (v_i, v_j) that play the role of links in this graph. A node can be any type of entity such as a person, a physical or abstract object and links are the representation of some kind of relation between nodes such as the fact that two people are friends.

Two nodes that are connected via a link are said to be adjacent. When a node is connected to itself, it is called a (self-)loop. In this paper, all networks (in the theory and in the experiments) are undirected, which means that there is no order in any pair of nodes connected (i.e. (v_i, v_j) is equivalent to (v_j, v_i)). Links can have several types of attributes associated which might be numerical, categorical or even complex (e.g. time series) [59]. In this paper, the only information that network ties may carry is a weight w_{ij}, which makes the corresponding networks "weighted". Weights are used to represent a degree of affinity (or similarity or closeness) between the nodes connected [20]. How affinity is then defined is dependant on each application. Graphs can be represented by a (n, n) adjacency matrix $\mathbf{A}$ that contains affinities a_{ij} with:

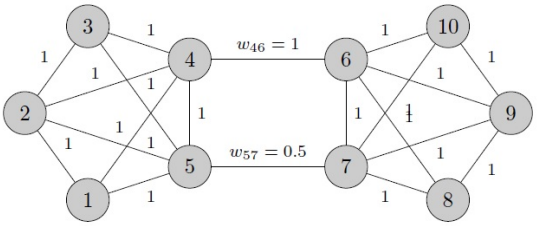
$$a_{ij} = \begin{cases} w_{ij} & if\ nodes\ i\ and\ j\ are\ connected \\ 0 & otherwise \end{cases} \tag{2.1}$$

where $a_{ij} = a_{ji}$ (i.e. $\mathbf{A}$ is symmetric) as we only use undirected graphs. In the case of an unweighted graph, w_{ij} is simply equals to 1 when nodes are connected and 0 otherwise. Figures 2.1a and 2.1b include visual representations of a graph and its adjacency matrix, respectively.

Another matrix representation of a graph is provided by the cost matrix $\mathbf{C}$ whose concept is opposite to the one of an adjacency matrix. Costs c_{ij} are related to the edges, are non negative and are sometimes calculated based on the affinities as $c_{ij} = 1/a_{ij}$ [20, 59]. For a weighted graph, the cost matrix is defined as:

$$\mathbf{C}_{ij} = \begin{cases} c_{ij} & if\ nodes\ i\ and\ j\ are\ connected \\ \infty & otherwise \end{cases} \tag{2.2}$$

The next notion is the concept of path or walk $\wp$ in a graph which is an ordered sequence of edges such that each node browsed is adjacent to the previous one on the path. A path connecting



(a) Visual representation of the graph

$$\mathbf{A} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0.5 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

(b) Adjacency matrix of the graph

Figure 2.1.: Example of a weighted graph. Source: [20]

nodes i and j is denoted by $\wp_{ij}$ and the set of all paths of different lengths possible in a graph G starting at node i and ending up at node j is denoted by $\mathcal{P}_{\text{ij}}$ [20, 59].

The degree of a node, d_i, corresponds to the number of incident edges or, in other words, the number of nodes adjacent to this node. The degree can be simply computed from the adjacency matrix as followed:

$$d_i = \sum_{j=1}^{n} a_{ij} = a_{i\bullet} \tag{2.3}$$

Then, $\mathbf{D}$ is the (n, n) "diagonal degree matrix" of the graph containing on its diagonal d_i in i-th position, while $\mathbf{d}$ is the vector containing all n degrees. The sum of all degrees of a graph is referred to as the volume of the graph and is denoted by $vol(G)$.

Finally, a useful matrix that will be used here is the (n, n) Laplacian matrix $\mathbf{L}$ which is defined for undirected graphs without self-loops as:

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{2.4}$$

This matrix is symmetric and positive semi definite [4, 9], a property that is important for us as it will be shown below.

3. Distances in a graph

In this chapter, we first introduce the concept of distance between nodes of a graph and the opposite concept, similarity. Then, we present different ways to capture (i.e. measure) distance in a graph, which leads us to what we called different types of distance. Distance measures will serve as a basis for clustering techniques to form a set of clusters for which the intra-cluster distance (similarity) is low (high) and at the same time the inter-cluster distance (similarity) is high (low).

3.1. Concept of distance and similarity

A distance between two nodes is used to measure their dissimilarity and is the opposite concept of a similarity measure. Let us start by similarity: such measure s shall respect the following properties [20]:

- $s(i, j) \geq 0$ where i and j are two nodes of a graph
- higher is $s(i, j) \geq 0$, more similar nodes i and j are
- s is symmetric, in other words $s(i, j) = s(j, i)$

Furthermore, intuitively, the maximum similarity should be attained when nodes are identical but this is not always the case as with inner product similarity that will be covered in section 4.

A dissimilarity index or distance $\triangle_{ij}$ should decrease in function of the closeness of the nodes and can be easily converted into similarity measure (and vice-versa) [20]. A distance measure follows the first and the third properties of a similarity measure and, in addition, shall respect the following ones[19]:

- $\triangle_{ij} = 0$ if and only if $i = j$
- $\triangle_{ij} \leq \triangle_{ik} + \triangle_{kj}$ (triangle inequality property)

Similarity (or distance) can be captured through two complementary sources: the features (i.e. attributes) of the nodes and the structure of the graph. More (less) two nodes share common features, more similar (distant) they are. The structure gives different types of information to measure similarity between two nodes: their proximity in the network, their degree of connection, whether they share common (sub-)structure and if they influence the graph in a similar way [20]. In this thesis, similarity will be derived from the second source only, from the structure of the graph.

3.2. Types of distances

From now on we talk in terms of distances until the introduction of kernels where we show how to convert distances into similarities. A good measure of distance should be able to capture the dissimilarity between the nodes. There are many types of distances measures between nodes of a graph but we only use five of them in this paper, all being part of the same family of distance.

The shortest path (introduced below but not used here) is maybe the most evident but it only takes into account the length of one path and not the connectivity of the nodes. In other words, it does not take into account all the other paths thus the fact that more paths they are between two nodes, less distant they are.

An alternative is the commute time distance that focuses on the connectivity but suffers from other limitations. Thus, we use also here a corrected version of this distance that will be experimentally tested compared to the original version. All that will be described in more details below.

The two distances already mentioned have some limitations and they lead to problems in some applications. As a consequence, recently, the research community started to look for distances interpolating those two and as a result the following ones were introduced: the logarithmic forest, free energy and randomized shortest path distances. They are all considered as part of the same family of distances together with the shortest path and the commute time distance. In other words, these 3 new distances can be viewed as located between the shortest path distance and the commute time distance representing the extremes of a continuum. Further, those new distances are even equal to these two extremes when their respective parameter takes a certain value as it will be explained below [20].

3.2.1. Shortest path

If we take a pair of nodes in the graph, they are likely to be connected through several paths of different lengths. The shortest path (also referred to as geodesic) is defined as the path for which the total cumulated cost along the path (derived from the cost matrix) is minimum. This total cost is then taken as the distance separating the nodes. In unweighted graphs, the shortest path is simply the most direct one (i.e. with the least number of edges) and the distance is simply equal to the number of edges separating the nodes. Some pairs may not be connected at all; then the graph is said to be disconnected (once there is one pair that is not connected) and the distance between these nodes is infinite or undefined. There are different methods to calculate this distance, see [20] for further information.

3.2.2. Markov chain and random walk

Following [33], Markov chains, an important branch of dynamic systems in discrete mathematics, are a system that evolves probabilistically. The concept comes from and has applications beyond graph theory. Here, it serves as foundation to describe the concept of random walk and then to define the commute time distance. A Markov chain is a finite process during which a marker (a walker in the case of graphs) jumps around among a finite set of states (nodes of the graph in this case)[1]. The system is said to evolve probabilistically because the transition from a node to another is determined according to probabilities [33]. The advantage of Markov chains is two fold: it provides a straightforward visualisation of a random walker and it has a simple matrix representation [20].

Suppose a connected graph[2] made up of n nodes. This graph has a set of n states possible associated which is denoted by $\{S_1, S_2, ..., S_n\}$. Markov chains are also determined by a set of transition probabilities p_{ij} which are associated to each edge (in the case of graphs). As an illustration, in the unweighted graph in figure 3.1, there are 5 states whatever is the starting point of the process. If vertex B is the starting node, A and C are the two next destinations possible and each corresponding edge has a (one step) transition probability of 1/2. Then, if C is the next state in the process, there are three possible destinations e_{CB}, e_{CD} and e_{CE} that all have associated (one step) transitions probabilities equal to 1/3. In other words, at each state, the probability to reach a non adjacent node as next state is 0 and the sum of all one step transition probabilities is 1.

[1] States can be locations or conditions, when Markov chains are used beyond graph theory

[2] In case of unconnected graphs they can be decomposed into several independent connected graphs as for their corresponding Markov chain [19]

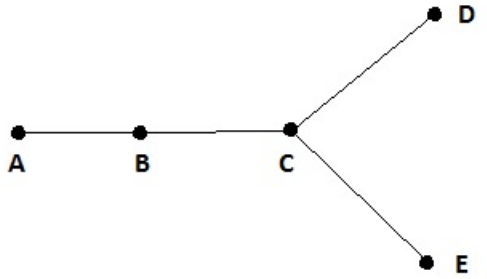


Figure 3.1.: Markov chain and random walk - simple graph

All in all, the marker (i.e. walker) of the system moves stepwise and randomly among the set of states (i.e. nodes). Furthermore, it is assumed that the probabilities in any states are independent from the past states, from the number of steps and from time; they only depend on the present state [20]. We define $\mathbf{P}$ as the matrix containing the one step transition probabilities and we call it the transition matrix. Its values are non-negative and the sum of any row is equal to 1 as the sum of all one step transition probabilities at each step in the process described above is 1 [20].

When the Markov chain concept is used to describe the sequence of nodes travelled by a random walker (i.e. when the marker is a walker and states are nodes), it is called a random walk on a graph. In weighted graphs, transition probabilities associated to each edge are proportional to the corresponding weights w_{ij}. As a consequence, one step transition probabilities can be easily computed from the adjacency matrix of weighted and unweighted graphs as followed:

$$p_{ij} = \frac{a_{ij}}{a_{i\bullet}} \text{ where } a_{i\bullet} \sum_{j=1}^{n} a_{ij} \tag{3.1}$$

Equation 3.1 means that, at any step, the random walker will choose which link to travel along randomly but still favouring the shortest links (i.e. the ones with the highest affinity a_{ij}), as transition probabilities are positively related to affinities. As a reminder, the walker does not take into account variables such as the previous links walked. In matrix form, $\mathbf{P}$ is computed like this:

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{A} \tag{3.2}$$

where is the diagonal degree matrix as defined above (see section 2.2). Finally, let us define a

node j as absorbing when it is impossible for a random walker to leave it. This is done by setting p_{jj} to 1 and all p_{ji} values to 0 (with $j \neq i$) [19, 23, 45].

3.2.3. Commute time distance

Based on the concept of Markov chain and the definition of transition probabilities, the average first passage time can be computed and then the average commute time that is an extension of the former.

The average first passage time $m(j \mid i)$ between two nodes j and i is defined as the average number of steps that a random walker starting at node $i \neq j$ will take to reach the ending node j for the first time (see [19]). Similarly, the average commute time $n(i, j)$ is the average number of steps taken by a random walker starting at $i \neq j$ to attain j for the first time and come back to starting node i. As a consequence, $n(i, j) = m(j \mid i) + m(i \mid j)$. As shown by [22, 30], the average commute time is a distance measure because it respects all properties of such measure outlined in section 3.1.

The average commute time distance is also called resistance distance for its close relationship with electrical networks [30, 55] and will be called simply "commute time distance" hereafter.

The commute time distances separating nodes of a graph can be computed from the Moore-Penrose pseudo-inverse[3] of the Laplacian matrix of the graph, denoted by $\mathbf{L}^+$. As [19] shows, the commute time distance that is a squared distance is then:

$$[\mathbf{\Delta}^{(2)}_{\mathrm{CT}}]_{ij} = n(i, j) = vol(G)\,(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\mathbf{L}^+(\mathbf{e}_i - \mathbf{e}_j) \tag{3.3}$$

where $\mathbf{L}^+$ plays the role of a covariance matrix and the basis vectors are embedded in $\Re^n$[20].

3.2.3.1. Corrected commute time distance

In [34], authors show that the commute time distance between two nodes i and j has a flaw. Indeed, when the size of the graph increases the distance converges to $\frac{1}{d_i} + \frac{1}{d_j}$ which is meaningless as distance measure. It has been observed empirically that the distance becomes quite small when the corresponding vertices are highly connected. As a solution, they provide a corrected version of the commute time distance where the pseudo-inverse Laplacian matrix in equation 3.3 is

[3] See for example [2]

replaced by $\mathbf{D}^{-1/2}\mathbf{L}^{+}\mathbf{D}^{-1/2}$, which is still positive semi definite (fore more information see [20]). The corrected commute time distance matrix is then:

$$\mathbf{\Delta}^{(2)}_{\mathrm{CCT}} = \mathbf{\Delta}^{(2)}_{\mathrm{CT}} - vol(G)(\mathbf{D}^{-1}\mathbf{e}\mathbf{e}^{\mathrm{T}} + \mathbf{e}\mathbf{e}^{\mathrm{T}}\mathbf{D}^{-1} + \mathbf{diag}(\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1})\mathbf{e}^{\mathrm{T}} + \mathbf{e}\mathbf{diag}(\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1})^{\mathrm{T}} - 2\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}) \tag{3.4}$$

Again, those distance measures are squared distances.

3.2.4. Logarithmic forest distance

Introduced by Chebotarev in [7], it is part of a class of parametric distances constructed on the matrix forest theorem [8]. Assume that G is an undirected weighted graph. Chebotarev first defined the following similarity matrix which is the regularized version of the Laplacian matrix:

$$\mathbf{K}_{\mathrm{RL}} = (\mathbf{I} + \alpha\mathbf{L})^{-1} \,\mathrm{with}\, \alpha > 0 \tag{3.5}$$

Then, he performed an element-wise logarithmic transformation as followed:

$$\mathbf{S} = \begin{cases} \gamma\,(\alpha - 1)\log_{\alpha}\mathbf{K}_{\mathrm{RL}} & \text{when } \alpha \neq 1 \\ \gamma\,\ln\mathbf{K}_{\mathrm{RL}} & \text{when } \alpha = 1 \end{cases} \tag{3.6}$$

where $\gamma > 0$ and α are parameters.

Finally, the similarity matrix $\mathbf{S}$ is transformed into the logarithmic forest distance matrix $\Delta^{(2)}_{\mathrm{LF}}$ that contains squared distances:

$$\mathbf{\Delta}^{(2)}_{\mathrm{LF}} = \mathbf{diag}(\mathbf{S})\mathbf{e}^{\mathrm{T}} + \mathbf{e}(\mathbf{diag}(\mathbf{S}))^{\mathrm{T}} - 2\mathbf{S} \tag{3.7}$$

Chebotarev proved that equation 3.7 is a distance matrix because his metric respects the properties of a distance that were mentioned in section 3.1.This new distance is equal to the shortest path distance when $\alpha \to 0^{+}$and to the commute time distance when $\alpha \to \infty$ [20, 29].

3.2.5. Free energy and randomized shortest path distances

3.2.5.1. The bag of paths framework and the bag-of-hitting-paths probabilities

The main idea behind the bag of paths framework is the probability that a path picked from a "bag of paths" has node i as starting node and j as destination node[4]. Each path is weighted according to its total cost so that low-cost paths (i.e. shortest paths) are more likely to be picked than high-cost paths (i.e. longest paths). This concept is the foundation of two distance measures studied in this thesis: the randomized shortest path and the free energy distance.

The bag of paths model uses the transition probability matrix $\mathbf{P}$ of the graph coming from the concept of random walks introduced above. In addition, it uses the cost matrix $\mathbf{C}$ containing the direct costs of the edges in order to compute the total cost of any path $\wp$. More specifically, this total cost is the sum of the direct costs over all edges along the path and is denoted by $\tilde{c}(\wp)$. It is important to note that it is better to set the costs independently from the adjacency matrix, that is, not computed from the affinities a_{ij} [16].

First, let us define a matrix $\mathbf{W}$ as:

$$\mathbf{W} = \mathbf{P} \circ \exp\left[-\theta \mathbf{C}\right] \tag{3.8}$$

where θ is a positive[5] parameter and $\circ$ is the element-wise (Hadamard) matrix product. Furthermore, the fundamental matrix $\mathbf{Z}$ is constructed as followed:

$$\mathbf{Z} = (\mathbf{I} - \mathbf{W})^{-1} \tag{3.9}$$

Then, [16] determines the Boltzmann probability distribution necessary to compute the probability to pick up a certain path up to a certain arbitrary length. From this distribution, they extended the concept to the probability of picking a path of any length starting in node i and ending in j. The bag-of-paths probability matrix $\mathbf{\Pi}$ that contains these probabilities for all pairs of nodes is then:

$$\mathbf{\Pi} = \frac{\mathbf{Z}}{z_{\bullet\bullet}} \tag{3.10}$$

[4] In this thesis, the version where zero-length paths are allowed is used
[5] $\theta > 0$

The real concept on which our two future distances are based is, however, the bag-of-hitting paths probabilities which is the same notion but where the ending node j can only appear at the end of the path. In other words, paths going through j twice or more are excluded from the bag. Technically, this is done by making node j as absorbing as in a Markov chain (see section 3.2.2). The bag-of-hitting-paths probability matrix $\mathbf{\Pi}_\text{h}$ is computed as followed:

$$\mathbf{\Pi} = \frac{\mathbf{Z}\mathbf{D}_\text{h}^{-1}}{\mathbf{e}^\text{T}\mathbf{Z}\mathbf{D}_\text{h}^{-1}\mathbf{e}} \,\text{with}\, \mathbf{D}_\text{h} = \mathbf{Diag}(\mathbf{Z}) \tag{3.11}$$

where $\mathbf{Z}\mathbf{D}_\text{h}^{-1}$ is the fundamental matrix of hitting paths denoted by $\mathbf{Z}_\text{h}$ [16, 20]. For more information, see [16, 20] that provides, among others, algorithms to compute those matrices, an adaptation of the model where zero-length paths are excluded and an intuitive interpretation of $\mathbf{Z}_\text{h}$.

3.2.5.2. Randomized shortest path distance

Based on the bag-of-hitting-paths probabilities, the *expected* cost of going from node i to node j is first computed and called the randomized shortest path *cost*. Then, the randomized shortest path distance between these nodes is the average of the randomized shortest path costs associated with going from i to j and coming back to i. The matrix containing the randomized shortest path distances for the whole graph is:

$$\mathbf{\Delta}_\text{RSP} = \frac{\mathbf{S} + \mathbf{S}^\text{T} - \mathbf{e}\left(\mathbf{diag}(\mathbf{S})\right)^\text{T} - \mathbf{diag}(\mathbf{S})\,\mathbf{e}^\text{T}}{2}, \text{ with } \mathbf{S} = \left(\mathbf{Z}\left(\mathbf{C} \circ \mathbf{W}\right)\mathbf{Z}\right) \div \mathbf{Z} \tag{3.12}$$

This distance is part of the same family than the logarithmic forest distance and therefore, when $\theta \to \infty$, this measure is equals to the shortest-path distance and, when $\theta \to 0^+$, it is equivalent to the commute time distance . A limitation of this metric is that it does not verify the triangle inequality property and is thus not a valid distance measure, as opposed to the logarithmic forest distance. Nevertheless, it will be considered as such because it has a nice interpretation and has generated good experimental results until now [20, 29, 56].

3.2.5.3. Free energy distance

The free energy distance (sometimes called potential distance) is a distance that also relies on the bag-of-hitting-paths probabilities. Again, it is part of the family of distances that includes

the previously mentioned distances and it is equal to the two extremes of the continuum, exactly as the randomized shortest path distance (same parameter, same values). It is computed as followed:

$$[\Delta_{\mathrm{FE}}]_{ij} = \begin{cases} \dfrac{\phi(i,j)+\phi(j,i)}{2} & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}, \text{ where } \phi(i,j) = -\frac{1}{\theta}\log z^{\mathrm{h}}_{ij} = -\frac{1}{\theta}\log\frac{z_{ij}}{z_{jj}} \tag{3.13}$$

As opposed to the randomized shortest path distance, this new distance respects the triangle inequality and is therefore a valid distance measure [16, 20, 29].

This chapter presented different types of distances, which allow us to capture in different ways the (dis)similarity between nodes of a graph. This is essential as we will try later to find communities (i.e. cluster) in graphs based on the (dis)similarity. In addition to comparing the quality of the different types of distances experimentally, we are interested to experiment the logarithmic forest distance because it has never been done to our knowledge. The next chapter is about kernels that provide a way to represent similarity. It is with similarity measures that clustering algorithms will directly work and those measures will be derived from the different types of distances.

4. Kernels on a graph

A kernel is a mathematical tool whose utility is to capture the similarity among a set of objects. In the case of graphs, we use it to represent the similarity between nodes of a graph. More specifically, here, we will use a kernel matrix $\mathbf{K}$ as input for the Ward's hierarchical clustering described in section 5.1. Element k_{ij} of this matrix is the similarity between node i and j. Different types of kernel matrices are used here, each one being derived from a certain type of distance (from the types defined in the previous section). Let us first introduce the concept of kernel in general.

A kernel is a function mapping two objects in a real number that represents a similarity. Assume a vector representation of each object in what is called an input space where each dimension is the measurement of a characteristic on the objects. The kernel transformation consists in an inner product between the vector representations of the nodes into an inner product space called the embedding space. Note that, in our case, we are working with graph structures and not with data defined in an Euclidean space thus the input space is not relevant. The kernel function can be formalised as followed:

$$k(i,j) \ : \ \Omega \times \Omega \rightarrow \mathbb{R} \text{ where } i \text{ and } j \text{ are the objects} \tag{4.1}$$

The output of the transformation in equation 4.1 can then be considered a similarity measure if it meaningfully represents the similarities between the two objects. In the case of kernels on graphs, the main advantage of using kernels is that it is able to capture some meaningful notion of similarity even though the objects cannot be represented with features.

One of the properties of a kernel function and matrix is that it is (should be) positive semi definite. Sometimes, a similarity matrix is called a kernel matrix even though it is not positive semi definite, but this is an abuse of language. In this thesis, we do this abuse of language with, for example, the free energy kernel (see section 4.1.2) and one of the research question is to test experimentally that the fact that it does not fulfil this property does not affect the quality of the clustering results (see section 10.1.3). The kernel matrix is also known to be symmetric. For more information about kernel functions and matrices, see [20, 48, 49].

Finally, there is a third space called the sample space which is an Euclidean space with as many dimensions as the number of data objects. This space will serve to perform the famous kernel trick or substitution [48] that is used in Ward's clustering (in section 5.1). Clustering algorithms like Ward's are usually designed originally for Euclidean distances so we need to adapt them in order to work with kernels.

Figure 4.1 is a way to visually represent a kernel matrix with a heat map where each colour represents a value from the matrix. In this case, blue colours are low values of similarity while colours close to red mean high similarity between the corresponding nodes. Note that, in the matrix underlying figure 4.1, nodes are ordered according to the "natural" classes.

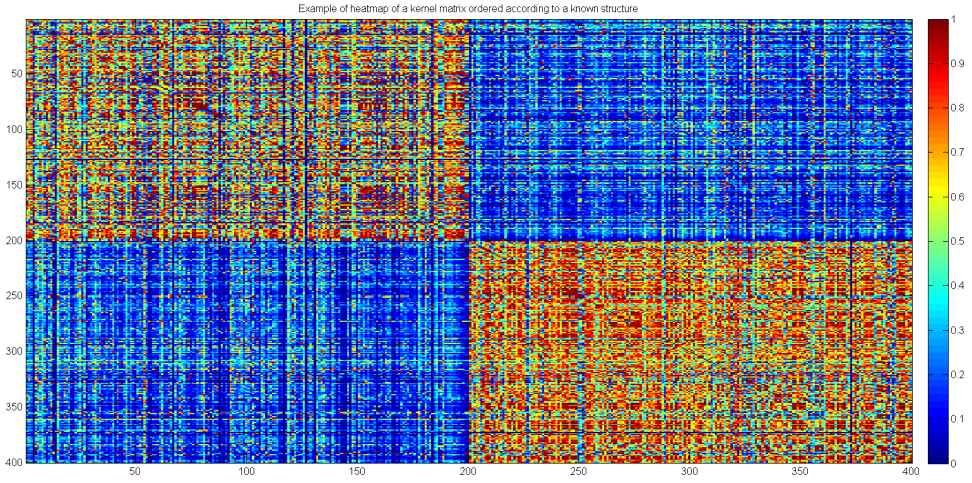


Figure 4.1.: Example of heat map for a kernel matrix ordered according to "natural" classes, for a graph of 400 nodes

4.1. Computing the kernels from the distance matrices

Multidimensional scaling shows that inner products (and kernels are inner products) can be computed easily from distances as followed:

$$\mathbf{K} = -\frac{1}{2}\mathbf{H}\boldsymbol{\Delta}^{(2)}\mathbf{H} \tag{4.2}$$

where $\mathbf{H} = (\mathbf{I} - \mathbf{e}\mathbf{e}^{\mathrm{T}}/n)$ is the centring matrix that removes in each column of the distance matrix the mean of the column. If the distances are not embeddable in an Euclidean space, then the kernel matrix is not positive semi definite [20, 27].

4.1.1. Sigmoid commute time (SCT) and sigmoid corrected commute time (SCCT) kernels

If we apply the transformation depicted in equation 4.2, we obtain after some steps that the commute time kernel is simply:

$$\mathbf{K}_{\mathrm{CT}} = \mathbf{L}^{+} \tag{4.3}$$

In other words, $\mathbf{L}^{+}$contains inner products between the node vectors in an Euclidean space where these node vectors are exactly separated by commute time distances. The commute time kernel is positive semi definite because the Laplacian matrix (thus its pseudo-inverse) respects this property [20].

To obtain the kernel from corrected commute time distances we apply the same equation 4.2 to the corrected distances and the end result is:

$$\mathbf{K}_{\mathrm{CCT}} = \mathbf{K}_{\mathrm{CT}} + \mathbf{H}\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}\mathbf{H} \tag{4.4}$$

Then, we apply to those two kernels an element wise sigmoid transformation and we obtain the corresponding sigmoid kernel $\mathbf{K}^{\mathrm{S}}$:

$$\left[\mathbf{K}^{\mathrm{S}}\right]_{ij} = \frac{1}{1 + \exp\left[-\alpha k_{ij}/\sigma\right]} \tag{4.5}$$

where k_{ij} is an element of the corresponding kernel matrix,α is a parameter and σ is the standard deviation of the elements of the kernel matrix. The sigmoid transformation converts all kernel matrix elements into a $[0-1]$ scale. This was done because we first experimented without a sigmoid transformation and it led to very poor results on some datasets. The commute time distance is embeddable in an Euclidean space so these two kernels are positive semi definite and valid kernels [19, 20, 30, 45, 55].

4.1.2. Logarithmic forest (LF), randomized shortest path (RSP) and free energy (FE) kernel

The logarithmic forest, the randomized shortest path and the free energy kernel, part of the same family of distances, are all computed through the same equation 4.2 where $\Delta^{(2)} = \mathbf{\Delta}^{(2)}_{\mathrm{LF}}$, $\mathbf{\Delta}^{(2)} = \mathbf{\Delta}^{(2)}_{\mathrm{RSP}}$ and $\Delta^{(2)} = \mathbf{\Delta}^{(2)}_{\mathrm{FE}}$, respectively.

They are not Euclidean distances therefore the similarity matrices are not positive semi definite and are not valid kernels but we will use the term kernel hereafter (by abuse of language) [29]. As a reminder, this led to one of the research question whose purpose is to verify that turning the free energy kernel into a positive semi definite matrix does not produce better clustering results. In [29], they observed that converting those "kernels" in positive semi definite matrices did not affect much the results of the kernel k-means, a clustering algorithm that is not studied here. To obtain a positive semi definite positive matrix, we simply order the eigenvalues (thus also the eigenvectors accordingly) and set the negative ones to 0.

This chapter presented the different types of kernel that were constructed following the same way but based on different types of distances. Now, the hierarchical clustering methods that we experiment in this thesis will be covered including the kernel-based Ward's clustering where kernels are used as input.

5. Hierarchical clustering methods

As mentioned earlier, we focus exclusively on clustering objects[1] based on their links in a graph or network (i.e. structure data) and not based on the observations of several features on the same objects (feature data). For instance, clustering a network of people extracted from Facebook is a potential application of this thesis but not segmenting the same people based on their age, sex and other variables. It is possible that a network has nodes having features like age or sex but it is not the case here.

The other particularity is that we compare hierarchical clustering algorithms only. Hierarchical clustering is a special type of clustering technique because it produces a hierarchy of clustering partitions, usually one for each number of clusters possible between 1 and n (with n nodes in the graph) [52]. There are two types of hierarchical algorithms, namely the top-down and the bottom-up ones, and the two methods studied here belong to the latter. Usually, such bottom-up technique (also called agglomerative) starts with the n nodes as n clusters and, iteratively, merges the two clusters most similar according to a certain criterion together. Please note that the Louvain method is not a traditional hierarchical method, it works a bit differently. Top-down (a.k.a. divisive) techniques work the other way around by starting with one clusters including all nodes and dividing iteratively the clusters.

Let us recall the graph G that has n vertices or nodes $V = \{v_1, v_2, ..., v_n\}$ and denote a clustering partition with m clusters by $C = \{C_1, ..., C_m\}$. Each cluster of a partition is a set of vertices $C_i \subseteq V$ such as clusters are disjoint $C_i \cap C_j = \emptyset$ for all $i \neq j$ and complete $\cup_{i=1}^{k} C_i = V$. In traditional bottom-up hierarchical clustering algorithms, at each step t, the partition created is said to be nested in the partition created at the previous step because the new clusters are subsets of the clustering partition produced at $t-1$. Typically, a hierarchical algorithm solution (i.e. the hierarchy of partitions) is illustrated visually by a dendrogram representing the nesting structure, which is a big advantage of such algorithms. An example is shown in figure 5.1 where clusters A and B form one of the partitions of the hierarchy. The vertical axis usually represents a similarity or dissimilarity metric that is measured at each step [52, 59].

[1] Again, the term "object" is used in a broad sense. It means as much humans as physical and abstract objects

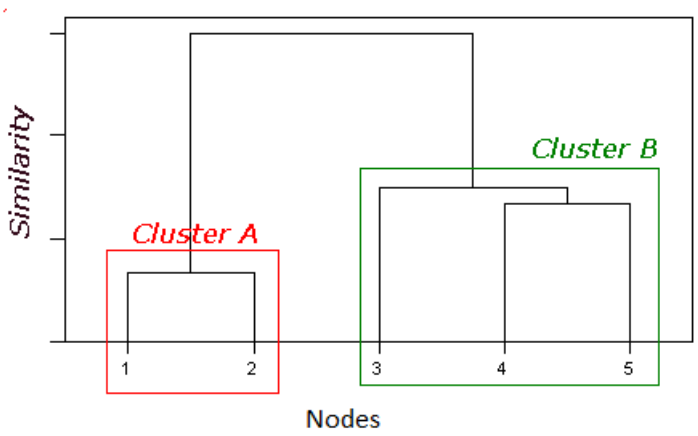


Figure 5.1.: Example of dendrogram for a hierarchical clustering on a 5-node graph

Alternatives to hierarchical clustering procedures are partitioning methods, density-based, model-based clustering and grid-based methods that may overlap, following the categorization made by [24]. Among the most popular, partitioning methods such as k-means start with a partition containing a pre-specified number of clusters k. Then, it iteratively changes the node allocation among the k clusters based on the optimization of a certain criterion [44].

5.1. Hierarchical clustering based on Ward criterion

Unsurprisingly, the Ward's bottom-up hierarchical clustering was originally based on a data matrix defined in an Euclidean space and on distances [53]. Let us first introduce the concept of the method by using this environment and then present the kernel-based version that will be used in this thesis for the experiments.

5.1.1. Distance-based version

Assume here n objects to be clustered that are represented by vectors $\mathbf{x}_i$ that are embedded in $\mathbb{R}^m$ where m is the number of features. The initial state of the algorithm is a partitioning where each object forms, alone, one cluster, in other words the partition is $\{C_k = \{\mathbf{x}_i\}, k = i = 1, ..., n\}$. Then, at each step, the two most similar or the least dissimilar clusters from the previous partition are merged until there remains only one cluster. Clustering often uses the notion of prototype which is a point representative of a cluster. With continuous attributes in an Euclidean space, the prototype is often a centroid such as the mean of all objects in the cluster while with categorical attributes it is often a medoid that is the most central data point of the cluster.

There are different versions of hierarchical clustering algorithms; they differ in the criterion that they use to measure this similarity/dissimilarity and to select the pair to merge. The Ward's hierarchical clustering uses the Ward criterion which is the *increase* in the sum of squared distances (SSE) as dissimilarity metric. In other words, the pair of clusters that leads to the minimum *increase* in SSE is merged, at each step. The SSE (also called variance or within-cluster inertia) within any cluster C_k is defined by:

$$SSE_k = \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \tag{5.1}$$

where $\boldsymbol{\mu}_k$ is the mean vector of all objects present in cluster C_k, in other words the prototype and centroid of cluster C_k . The Ward criterion is the net change in SSE when two clusters C_k and C_l are merged into C_{kl} which is simply $\triangle SSE_{kl} = SSE_{kl}(after\, merge) - SSE_{kl}(before\, merge) = SSE_{kl} - (SSE_k + SSE_l)$. After several computational steps, [59] shows that this net change can be rewritten as:

$$\triangle SSE_{kl} = \left(\frac{n_k n_l}{n_k + n_l}\right) \|\boldsymbol{\mu}_k - \boldsymbol{\mu}_l\|^2 \tag{5.2}$$

where n_k is the number of objects contained in cluster C_k [20, 52, 55, 59].

5.1.2. Kernel-based version

To apply Ward's clustering on graphs, we use kernels on graph because it allows us to work around the fact that we lack an Euclidean space and thus we cannot compute mean vectors[2]. Let us assume that we have a kernel $\mathbf{K}$ on a graph, that can be any of the types described in section 4.1. The nodes are represented by node vectors $\mathbf{x}_i$ and the prototypes by vectors $\mathbf{g}_k$ (where k is the cluster C_k) in an Euclidean embedding space (also called feature space).

Initially, we have in the feature space $\mathbf{g}_k = \mathbf{x}_k$ for $k \in \{1, ..., n\}$ because each node forms alone one cluster so the prototypes are equivalent to the nodes. We use here the kernel trick mentioned previously that consists in using $\mathbf{X}^{\mathrm{T}}\mathbf{h}_k = \mathbf{x}_k$ in order to express the prototype vectors in the sample space where they are denoted by $\mathbf{h}_k$. If you pre-multiply this equation by $\mathbf{X}$, knowing that $\mathbf{K} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$ (as kernels are inner products), we obtain $\mathbf{K}\mathbf{h}_k = \mathbf{k}_k = \mathbf{K}\mathbf{e}_k$. Therefore, the prototypes vectors of the n initial clusters are :

[2] An alternative way would be to work with a distance matrix such as the ones introduced in section 3.2 but this goes beyond this thesis [14]

$$\mathbf{h}_k = \mathbf{e}_k \tag{5.3}$$

At each step, after each merge of two clusters C_k and C_l into $C_{k\cup l}$, the new prototype vector (i.e. of the newly created clusters) in the embedding space is simply a weighted average of the former vectors:

$$\mathbf{g}_{k\cup l} = \frac{n_k\mathbf{g}_k + n_l\mathbf{g}_l}{n_k + n_l} \tag{5.4}$$

Using the same kernel trick, the new prototype vector in the sample space is:

$$\begin{cases} \mathbf{h}_{k\cup l} = \frac{n_k\mathbf{h}_k + n_l\mathbf{h}_l}{n_k + n_l} \\ n_{k\cup l} = n_k + n_l \end{cases} \tag{5.5}$$

From equation 5.5, we can verify that if $\mathbf{h}_k^{\mathrm{T}}\mathbf{e} = 1$ and $\mathbf{h}_l^{\mathrm{T}}\mathbf{e} = 1$ then $\mathbf{h}_{k\cup l}^{\mathrm{T}}\mathbf{e} = 1$. Further, if it is initially true, it remains so during the whole clustering process. Finally, it can also be verified with the same equation that $[\mathbf{h}_k]_i = 1/n_k$ at any step. In other words, each element of any prototype vector $\mathbf{h}_k$ simply equals to $1/n_k$ if this element belongs to the corresponding cluster k and 0 otherwise.

Now let us define the Ward criterion which is the increase in within-cluster inertia brought by the pair of clusters merged. Similarly to equation 5.1, we can define the total within-cluster inertia J (i.e. total SSE) in the embedding space of a partition with m clusters (that can come from any step of the algorithm):

$$J(\mathbf{g}_1, ..., \mathbf{g}_m) = \sum_{k=1}^{m} SSE_k = \sum_{k=1}^{m} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \mathbf{g}_k\|^2 \tag{5.6}$$

The increase in within-cluster inertia resulting from a merge of cluster C_k and C_l is then obviously$\triangle J(C_k, C_l) = J(after\, merge) - J(before\, merge)$ as we did above in the distance-based version. From equation5.2, we deduce that this quantity is:

$$\triangle J(C_k, C_l) = \triangle SSE_{kl} = \left(\frac{n_k n_l}{n_k + n_l}\right) \|\mathbf{g}_k - \mathbf{g}_l\|^2 \tag{5.7}$$

If we apply the kernel trick, this gives in the sample space:

$$\triangle J(C_k, C_l) = \left(\frac{n_k n_l}{n_k + n_l} \right) (\mathbf{h}_k - \mathbf{h}_l)^{\mathrm{T}} \mathbf{K} (\mathbf{h}_k - \mathbf{h}_l) \tag{5.8}$$

Therefore, at each step, the pair of clusters (C_k, C_l) for which this quantity is minimum is merged. Meanwhile, the corresponding prototype vectors and cluster sizes are updated according to equation 5.5. This procedure is repeated until there is only one cluster remaining [20, 55].

5.2. Louvain method based on modularity

We first introduce the modularity measure that is then used as an objective criterion to maximise in the Louvain method to identify communities in networks.

5.2.1. Modularity

Modularity was introduced by [40] to quantify the quality of a particular division of a network. Suppose a particular partition of a graph G into k communities (i.e. clusters) ; and $\mathbf{E}$ as the (k, k) symmetric matrix whose element e_{ij} is the fraction of all edges in the network that link nodes in community i to nodes in community j (according to the partition supposed).

Its trace, $trace(\mathbf{E})$, gives the fraction of edges in the network that connect vertices among the same community. Obviously, higher is this scalar, better is the partition. However, it is not a good indicator of the quality of the partition because a partition with $k = 1$ cluster automatically means that the maximum value of this indicator is reached because $trace(\mathbf{E}) = 1$.

As a consequence, the authors go further by defining a new network where the edges fall between vertices randomly without regard for their respective communities, which means that $e_{ij} = e_{i\bullet} e_{\bullet j}$. The modularity Q of the original network G is then the fraction of within-community edges (defined above as $trace(\mathbf{E})$) minus the expected value of the same quantity (i.e. the fraction of within-community edges) in our new network with random connections. According to their notation:

$$Q = \sum_i (e_{ii} - e_{i\bullet}^2) \tag{5.9}$$

The total modularity Q for a given partition of the graph G is a scalar ranging from -1 to 1 but in practice values range from 0,3 to 0,7. We now define the equivalent of equation 5.9 with the

notation that is used to introduce the Louvain method and that leverages the adjacency matrix **A** of the graph:

$$Q = \frac{1}{vol(G)} \sum_{i,j \in G} \left[a_{ij} - \frac{a_{i\bullet} a_{j\bullet}}{vol(G)} \right] \delta(\ell_i, \ell_j) \tag{5.10}$$

where ℓ_i is the cluster label of node i and the function $\delta(\ell_i, \ell_j)$ is 1 if $\ell_i = \ell_j$ and 0 otherwise.

5.2.2. Louvain algorithm

The Louvain method, originally introduced in [3] as a method to find communities (i.e. clusters) in a graph, is also a bottom-up clustering procedure. Though, it is a bit different from traditional hierarchical clustering like Ward's method. The first difference is that it uses modularity as criterion to maximise in order to decide which pair of clusters to merge[3] with the objective to produce partitions with high-modularity. In large graphs, computing the modularity is computationally hard so approximation algorithms like the Louvain method are used, which is one of the main advantages of the method. The fastest of these algorithms for large graphs that existed before the Louvain method had several limitations therefore the latter was introduced [3].

The second important difference with Ward's clustering is that, though it also starts with the n nodes as n clusters, the Louvain method sometimes merges more than one pair of clusters per step. In fact, the Louvain method works with "passes" that are iterations repeated until there is no further improvement in modularity. It means that the method does not especially end with one big cluster regrouping all the nodes (third difference). Each pass always involves the following two steps:

1. **Local optimization:** At each iteration of this step, we compute the changes in modularity that would occur if we move a certain node i to its neighbouring clusters (i.e. clusters different from the present one that includes at least one adjacent node of i). The formula for computing this change in modularity is provided at the end of this section. Then, the node is moved to the cluster leading to the largest increase in modularity but only if it is an increase in modularity, otherwise it does not move. All the nodes are considered one by one without exception and it is done in random order. When all n nodes have been considered, the procedure continues in a new random order. In other words, it reconsiders all n nodes in a new random order, with the same procedure. It only stops when all n nodes *in a row* stay in their respective cluster because modularity cannot be increased for *none* of them.

[3] This same measure is sometimes used to compare the quality of clustering partitions [37]

2. **Node aggregation or coarsening:** the first step considered each node individually but is not the case anymore here. Instead, during this step, a new graph G' is built with the clusters produced at the end of step 1 as nodes of G'. The affinity between two nodes of G' is then equal to the sum of the affinities of all pairs of nodes that bridge the two related clusters in G (related to the two nodes of G'). Also, each node of G' has a self-loop with a weight equal to the sum of the weights of all pair of nodes that the corresponding cluster of G included. The adjacency matrix of the new graph G' is then $\mathbf{A}' = \mathbf{U}^{\mathrm{T}}\mathbf{A}\mathbf{U}$ where $\mathbf{U}$ is the membership matrix of G with elements $u_{ik} = 1$ if node i belongs to cluster k and 0 otherwise. Finally, step 1 is performed on the graph G', which is the beginning of a new "pass".

At each "pass", the number of clusters decreases until no improvement in modularity is made after step 1 *and* step 2. Then, the algorithm stops and the number of clusters of the final partition is the optimal number of clusters.

Before computing the increase in modularity coming from moving a node to another cluster, let us first define the total modularity associated with a certain partition, the equivalent of equation 5.10 but with matrices:

$$Q(\mathbf{u}_1, ..., \mathbf{u}_m) = \frac{1}{vol(G)} \sum_{k=1}^{m} \mathbf{u}_k^{\mathrm{T}} \mathbf{Q} \mathbf{u}_k \tag{5.11}$$

where $\mathbf{Q}$ is the modularity matrix, $\mathbf{Q} = (\mathbf{A} - \frac{\mathbf{d}\mathbf{d}^{\mathrm{T}}}{vol(G)})$ and $\mathbf{u}_k$ is the k-th column of $\mathbf{U}$.

Now, as [20] shows, the change in modularity due to a node i moving to a cluster C_l is:

$$\triangle Q(i, C_l) = \frac{2}{vol(G)} (\mathbf{e}_i + \mathbf{u}_l - \mathbf{u}_k)^{\mathrm{T}} \mathbf{q}_i \tag{5.12}$$

where $\mathbf{q}_i = \mathbf{Q}\mathbf{e}_i$ is the i-th column of $\mathbf{Q}$. The algorithm only computes the quantity from equation 5.12 and it is quite efficient because, for each node i, only $\mathbf{u}_l^T \mathbf{q}_i$ needs to be computed for each and every target cluster C_l [20].

This chapter presented the two clustering algorithms that are implemented and compared in the experiments. As opposed to the Louvain method, the Ward's hierarchical clustering produces several partitions, one for each number of clusters between 1 and n. As a consequence, we are now interested in ways to select the best partition out of the hierarchy, in other words to select the "best" number of clusters. The purpose of the next chapter is to present several methods able to achieve that goal and present in details the one that is used in this thesis, the L method.

6. Approaching the "natural" number of clusters

Several clustering methods suffer from the limitation that the number of clusters has to be specified by a human. Non hierarchical procedures usually require to specify this number as an input parameter while (traditional) hierarchical algorithms like Ward's clustering produce a series of partition, one for each number of clusters from 1 to n. In both cases, it requires this person either to have specific knowledge about the application or to try different numbers by trial and error, which is not always practical. To fill this gap, we are thus interested in procedures that can determine an optimal number of clusters. In our case, this number will help us to keep only one partition from the hierarchy of partitions produced by the Ward's clustering introduced in the previous chapter. First, we review the literature on the topic by presenting some procedures and then we explain the L method that is used in this thesis.

6.1. Presentation of different procedures

There are several procedures that are able to heuristically approach the "natural" number of clusters (i.e. determining an optimal number of clusters) based on a hierarchy of clustering partitions. The performance of 30 of these procedures (referred to as stopping rules) was compared in [36]. As an illustration, we briefly present the three best rules according to the experiments made by the very same [36]. Note two things: these rules are not specific to graph clustering and they were sometimes applied on a part of the hierarchy to eliminate distracting patterns.

1. Calinski and Harabasz index [5]: is computed for each level of the hierarchy (i.e. each number of clusters) as $[\text{trace}(\mathbf{B})/(k-1)] \,/\, [\text{trace}(\mathbf{W})/(n-k)]$ where n and k are the total number of nodes and the number of clusters in the solution, respectively. The $\mathbf{B}$ and $\mathbf{W}$ matrices contain the between and pooled within cluster sum of squares and cross products. The number of clusters associated with the maximum value of this index is then considered as the optimal number of clusters.

2. $Je(2)/Je(1)$: Duda and Hart introduced in [12] this ratio criterion where $Je(2)$ is the sum of squared errors within the two clusters just before their merge and $Je(1)$ the squared errors when they form one cluster. This criterion is computed for each number of clusters (i.e. each merge) in an ascending order. Each time, an hypothesis test is performed which consists in rejecting the hypothesis of one cluster if the ratio is smaller than a certain critical value. When the hypothesis is first rejected, the corresponding number of clusters (i.e. before the merge) is then considered as optimal.

3. C-Index: this index was reviewed in [26] and is computed as $[SSE - S_{min}] \,/\, [S_{max} - S_{min}]$[1]. If m is the number of pairs of objects in the same cluster, S_{min} and S_{max} are the sum of the m smallest and largest distances between nodes, respectively, among all nodes of the dataset. The C-index is comprised between 0 and 1 and the minimum value indicates the optimal number of clusters.

In [6], another rule was recently proposed and called "Framework for Optimal Selection of Clusters" (FOCS). Originally, FOCS was introduced within a semi-supervised context (where some items are constrained to be in the same cluster) but the authors also presented an adaptation for unsupervised clustering applications. Actually, the idea behind the unsupervised version is quite straightforward. First, a measure of cluster quality is chosen, let it be $S(C_i)$ where C_i is a cluster. Then, the sum of the qualities over all clusters of a partition p is the objective function that we denote by J in equation 6.1.

$$J_p = \sum_{k=1}^{m} S(C_k) \tag{6.1}$$

Then, the optimal number of clusters is the one corresponding to the partition that reaches the maximum value of this function among all partitions. The framework is suited to any unsupervised measure of cluster quality but the original paper decided to experiment their method with cluster lifetime as measure of cluster quality. The lifetime of a cluster is defined as the length of the dendrogram scale along the levels in the hierarchy where this cluster exists [28].

Finally, another procedure to achieve the same goal is the L method. In this thesis, we have decided to focus on this method because it seems promising. It is described in the following section.

[1] SSE is the sum of within cluster inertia over all clusters

6.2. The L method

6.2.1. Introduction

The L method[2] was originally introduced in [46] and is presented as an efficient algorithm to determine the optimal number of clusters. To do so, this technique locates the "knee" - that is the point of maximum curvature - in an evaluation graph that can be derived from the solution of hierarchical clustering algorithms. The x-axis of such evaluation graph is the number of clusters and the y-axis is the value of the evaluation function used by those algorithms to measure what a "good" cluster is. For instance, the evaluation function of Ward's clustering (on which we apply later the L method) is the increase in within-cluster inertia coming from each merge. Figure 6.1 represents a typical evaluation graph.

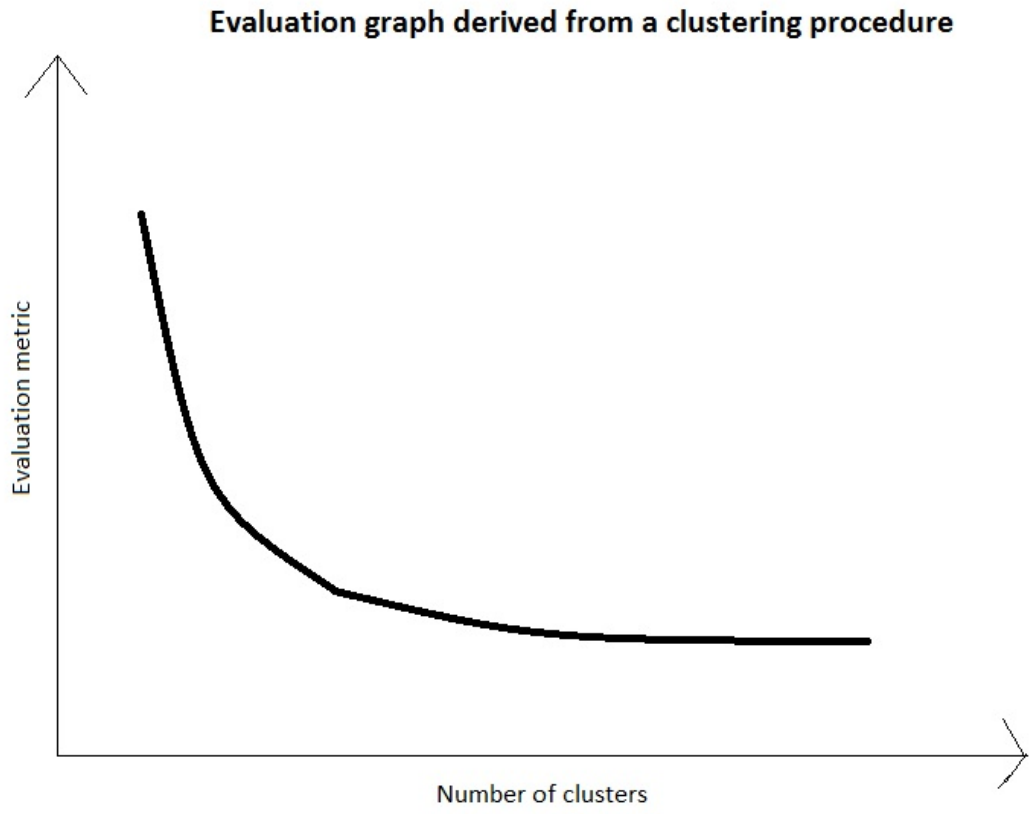


Figure 6.1.: Example of evaluation graph that can be derived from clustering procedures

Intuitively, the knee is considered as the optimal point because it is the point that separates the right-side of the graph where y-values are low and increase slowly, from the left-side where they

[2] It should not be mixed up with the Louvain method, they have no relation.

increase very rapidly. In other words, on the right side, very similar clusters are being merged because the evaluation metric is low while on the left side the steep increase in the metric means that very dissimilar clusters are being grouped together. Main clustering theories (e.g. [51]) explain that a "good" clustering solution is a situation where the objects (nodes in our case) within a cluster are very similar to each other while the clusters are very dissimilar between each other. Therefore, finding the knee means finding this balance, that is the number of clusters for which similar clusters are merged together but where dissimilar clusters are kept separated.

There are other procedures that try to find the knee, some of them are local and others like the L method are global in the sense that they consider all the points of the graph to make a decision. The L method has the benefit of being global thus less sensitive to outliers that might prevent the true knee to be located. In addition, it works well even with curves including abrupt jumps.

In [46], the L method was shown to be both much more effective in approaching the "natural" number of clusters and much more time-efficient than the Gap Statistic, another way to estimate the "natural" number of clusters. This number of clusters was found 10 out of 12 times by the L method against 3 out of 17 times by the Gap Statistic. The former took a fraction of second for all tests, while the latter took tens of minutes. According to [46], the other algorithms designed to determine the optimal number of clusters are inefficient.

Another advantage of the L method is that it works with all hierarchical algorithms that produces an evaluation graph, whatever is the evaluation function used by the algorithm. The evaluation metric can be of any type (e.g. distance, similarity, error or quality) and can be either global or greedy measures. Greedy or local measures are metrics computed based on a part of the dataset, such as the two clusters being merged or split by the clustering algorithm. Nevertheless, the L method turned out not working as well with global measures because the knees in the graphs were not as pronounced [46].

6.2.2. Core process

In the original paper [55], the evaluation metric used was the Ward's criterion, the increase in within-cluster inertia due to the merge performed based on the partition with x clusters. In order to find the knee, the original L method fits two straight lines, for each possible number of clusters (i.e. x value) from 3 to $(n-2)$ [3], one line on the left of this value and one on the right. For each number of clusters, the total root mean squared error (RMSE) of the two fittings is computed[4]. Then, the location of the knee is the number of clusters for which this sum is

[3] The reason for that is that the original paper[46] uses an evaluation graph starting at x = 2 and fitting requires each of the two fitted lines to contain at least two points

[4] This is the sum of the RMSE of each fitting weighted with the number of plots covered by the fitting

the lowest. The process is illustrated in figure 6.2 below that represents the method on a small evaluation graph. The knee is located at $x = 3$ because the fitting in the lower-left picture looks like the best and total RMSE the lowest.

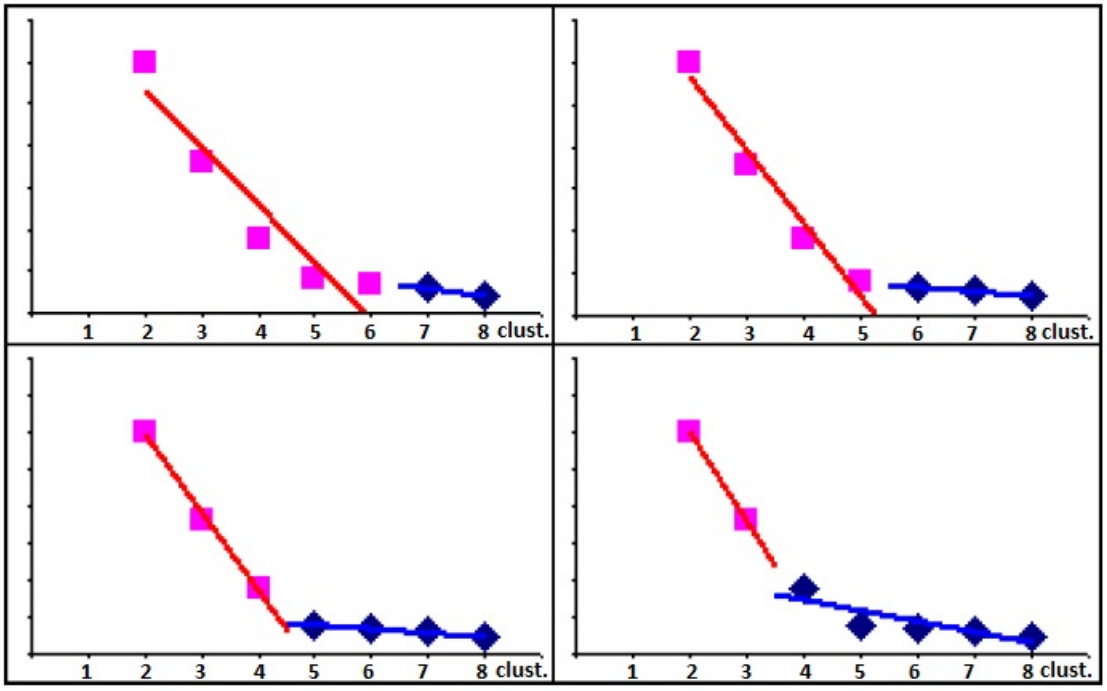


Figure 6.2.: L method - all four possible pairs of best-fit lines for a small evaluation graph. Source: [46]

An issue that arose in this research is that the minimum x that this method is able to find is 3 and several of the datasets used here have 2 "natural" classes. It only starts at 3 because the Ward Criterion is only computable until $x = 2$ clusters and the L method needs at least two points to fit a line on the left. Therefore, we used, mainly, another metric for the y-axis of the evaluation graph, the total within-cluster inertia at x clusters, which is a global measure. In this case, the evaluation graph starts at $x = 1$ and the first number of clusters that can be found by the L method is 2. Still, in the experiments, we ran the L method with the original evaluation metric to test the difference between the two metrics.

6.2.3. Refinements

The fitting procedure was the core algorithm of the method, then the algorithm proceeds with refinements. The reason is that, in large graphs (i.e. large x), many values are irrelevant and having them prevents from having an "L shaped curve" thus preventing the algorithm from

finding the best knee [46]. As a consequence, refining consists in defining the maximum value[5] of the graph as the double of the knee previously found and running the core algorithm again until the knee does not change.

This chapter showed that there are many procedures aimed to determine an optimal number of clusters from a hierarchy of clustering partitions. They vary pretty much in how they reach that objective and all of them can be applied on all types of data structure thus not only on graphs. The L method that apparently gave good results on a Ward's clustering hierarchy is the one tested in the experiments. Its quality will be compared to the optimal number of clusters found by the Louvain method, through the "natural" number of clusters as benchmark.

[5] This maximum value cannot be lower than 20 because there needs to be a reasonable number of point in order to avoid detecting false trends

Part II.

Experiments

7. Description of the datasets

In this chapter, we present the 16 datasets on which clustering algorithms are experimentally tested and compared. They are mostly social networks and their size is relatively limited, the biggest graph containing 6142 nodes. We did not have the opportunity to make the experiments on more datasets because network datasets including labels identifying the "natural" classes of the nodes are rare.

Newsgroup datasets: those datasets[1] are based on a collection of unstructured documents coming from several discussion groups of the Usernet diffusion list. Nine subsets were extracted from this collection and, in each one, the documents have been classified by topic. The topics - that are the "natural" classes in this case - contain 200 documents each (see Appendix A for further information about the topics). Three of the newsgroup subsets used contain two topics or classes (i.e. 400 documents), three contain three topics (i.e. 600 documents) and the three remaining subsets contain 5 different topics (i.e. 1000 documents). The original data have been pre-processed as described in [55] to reduce the high-dimensionality of the feature space. The weights of the links between the documents of the network depends on the terms that they have in common.

Political books dataset: the political books dataset refers to an unweighted network of books about US politics and was compiled by Victor Krebs[2]. The edges indicate how frequent two books were acquired by the same buyers on Amazon. The 105 books are grouped into 3 classes depending on their political orientation (conservative, liberal, or neutral)[38].

Zachary dataset: this network represents friendships between the 34 members of a karate club at a US university, as described by [58]. Members belong to one of the two sub-groups that our clustering methods will try to predict. The network is represented by an unweighted graph where two nodes are connected if they consistently interacted in the context outside of the club.

LFR datasets: those datasets represent binary networks of 600 vertices that were generated via the LFR benchmark as introduced by [31]. In this paper, two variants are used:

[1] Available here: http://qwone.com/~jason/20Newsgroups/
[2] Available on his website: http://www.orgnet.com/

- LFR1: it has 6142 edges and includes 3 classes
- LFR3: it has 5233 edges and includes 6 classes

School dataset: this dataset represents a high school with 236 students and teachers, the nodes of the network. There are two variants of this dataset:

- one with the two classes of each year kept separated, thus containing 11 groups in total
- the other with the classes of the same education year grouped together thus containing 6 groups in total

Football dataset: this last dataset [3] is a representation of American football games that took place during the 2000 season. Nodes in the unweighted graph are teams of the college Division I and the links refer to whether two teams played against each other. The 115 teams are divided into 12 conferences (i.e. the "natural" classes), each one containing around 8 to 12 teams. Games involving teams from the same conference are more frequent than inter-conference games. The number of inter-conference games played by the teams depend positively on their geographic proximity. This network is interesting because the system of conferences is a known community structure [21].

Table 7.1 provides an overview of the datasets.

[3] Available here: https://networkdata.ics.uci.edu/data.php?id=5

Dataset family	Dataset name	Number of nodes	"Natural" number of classes
Newsgroup	news_2cl_1	**400**	**2**
	news_2cl_2	**400**	**2**
	news_2cl_3	**400**	**2**
	news_3cl_1	**600**	**3**
	news_3cl_2	**600**	**3**
	news_3cl_3	**600**	**3**
	news_5cl_1	**1000**	**5**
	news_5cl_2	**1000**	**5**
	news_5cl_3	**1000**	**5**
Political books	polbooks	**105**	**3**
Zachary	zachary	**34**	**2**
LFR	LFR1	**6142**	**3**
	LFR3	**5233**	**6**
School	school_6	**236**	**6**
	school_11	**236**	**11**
Football	foot	**115**	**12**

Table 7.1.: Overview of the datasets used in the experiments

8. Quality measures

In this section, we introduce the 3 metrics that are used to measure the quality of the clustering partitions generated by the clustering algorithms. More specifically, these measures are used for comparing both (the partitions given by) the different kernels between each other (first research question) and the clustering algorithms (second part of the second research question). All 3 are external quality measures that judge the quality of a clustering partition given by a clustering algorithm with a "natural" partition with classes determined by human judgement or with the context. Higher those measures are, better will be the clustering partition assessed. On the other hand, there exists internal quality measures such as the ones used in evaluation functions of clustering algorithms but they are not always effective in an application [35].

8.1. Correct classification rate (CCR)

The (correct) classification rate is simply the percentage of objects correctly classified. This type of measure presents serious limitations such as the fact that it evaluates how well the clustering algorithm has labelled the clusters according to the original class labels. This can lead to poor results because the cluster labels may be switched even if the classes were well identified [47].

8.2. Normalized mutual information (NMI)

The normalized mutual information (NMI) is a measure ranging from 0 to 1 that indicates how much information is shared between the "natural" classes and the clusters found by the algorithm. The normalization ensures that the measure does not depend on the number of clusters, thus allowing to compare solutions with different number of clusters. [35] provides further details about this metric and its formula.

8.3. Adjusted rand index (ARI)

The computation of this metric is based on the evaluation of each of the $\frac{N(N-1)}{2}$ pairs of nodes. The rationale behind this measure is that the quality depends on how many pairs are correctly classified as similar or dissimilar . As it evaluates the relation between the objects and not the labels produced by the algorithm, it does not have the same issue as the classification rate because it evaluates how well the algorithm split the objects. There are four cases to which each pair can belong. The two first ones are true positive (TP) when the two nodes are correctly grouped into the same cluster by the algorithm and true negative (TN) when they belong to different classes and they are correctly put in different clusters. The two last scenarios are the false positive (FP) and the false negative (FN) when they are incorrectly grouped into the same clusters and incorrectly kept separated, respectively. The rand index RI is then the percentage of correct allocations:

$$\mathrm{RI} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}} \tag{8.1}$$

An issue with the rand index is that its expected value for two random partitions (in this case the partitions with the classes and the clusters) does not take a constant value [57]. To correct that, [25] suggested to make the adjustment in equation 8.2 where the index is RI. We obtain the adjusted rand index that equals 0 when the rand index equals its expected value and 1 as a maximum.

$$\text{adjusted index} = \frac{\text{index} - \text{expected index}}{\text{maximum index} - \text{expected index}} \tag{8.2}$$

9. Experimental procedure

9.1. Parameter tuning

In order to build the free energy distance (thus kernel) that is used in the Ward's clustering, the first step is to optimize its parameter θ (see section 3.2.5.3 for the theory). On one selected dataset[1], we applied the clustering algorithm on free energy kernels corresponding to different values of θ. Then, the value of θ giving the highest value of a certain quality measure (in our case the NMI) is then used for all subsequent experiments, on all datasets. We assume here that the parameter stays optimized for all datasets. Remember that Ward's clustering produces a hierarchy of partitions for every number of clusters. In order to simplify the tuning, we chose to only take into consideration the clustering partition with the "natural" number of clusters and compare them across several values of θ. As table 9.1 shows, $\theta = 0,01$ yields the highest NMI and will be thus used for the subsequent analysis.

θ	0,000001	0,01	0,1	1	10	20
NMI score	**0,0317**	**0,4032**	**0,3453**	**0,4006**	**0,3509**	**0,3813**

Table 9.1.: Free energy distance - tuning results of θ

The exact same procedure was used to tune the parameter α of the logarithmic forest distance (see 3.2.4 for the theory). Results given in table 9.2 show that $\alpha = 1$ is the optimal value because the corresponding NMI is the highest, so this value will be used hereafter.

α	0,000001	0,01	0,1	1	5	10	20
NMI score	**0,2340**	**0,3593**	**0,3608**	**0,3929**	**0,3285**	**0,1203**	**0,0809**

Table 9.2.: Logarithmic forest distance - tuning results of α

Table 9.3 summarizes for all kernels the parameter values that they use, including the ones that were already optimized. Apart from the free energy and logarithmic forest kernels, the parameter values are the ones commonly used in the literature. As a reminder, the SCCT and SCT kernels

[1] news_3_parameterTuning provided by M. Saerens, it comes from the newsgroup dataset and was used only for parameter tuning

need a parameter α for the sigmoid transformation while the others need it to compute the underlying distance.

Type of distance / kernel	Parameter value
Sigmoid transformation on kernels	$\alpha = 7$
Free energy distance	$\theta = 0,01$
Randomized shortest path distance	$\theta = 0,01$
Logarithmic forest distance	$\alpha = 1$ and $\gamma = \ln(\exp(1) + \alpha^{(2/n)})$ [2]

Table 9.3.: Overview of parameter values used to compute the different types of distance / kernel

9.2. Clustering procedure

When any of the clustering algorithms is launched, it computes the kernel (or the modularity matrix in the case of the Louvain method) by taking as input the adjacency matrix of the graph and the tuned parameter value. The output of the Ward's clustering algorithm is a series of vectors, one for each number of clusters from 1 to n, including cluster assignments of the nodes. Meanwhile, the only meaningful output of the Louvain method is the same vector but only for the number of clusters it considers as optimal (i.e. the last partition, when the algorithm stops). See sections 5.1 and 5.2.2 for the theory. This difference has the following implications for the experimental comparisons:

First research question: because this question compares only Ward's clustering solutions between each other, partitions for the "natural" number of clusters and for the optimal number of clusters are available. Therefore, two comparisons are made in each sub-question: one is to compare the different kernels based on the corresponding clustering partitions for the "natural" number of clusters and the other is to compare based on the partitions for the optimal number of clusters.

Second research question: as the Louvain method does not automatically give a partition for the "natural" number of clusters, only the clustering partition for the optimal number of clusters is used for the comparison.

For the Ward's clustering, the optimal number of clusters is determined by the L method based on the Ward's output hierarchy (see section 6.2 for the theory).

Finally, note that the Ward's hierarchical clustering is deterministic and was thus run once. However, the result of the Louvain method changes at each iteration because it is affected by the order of the nodes in the adjacency matrix. As a consequence, at each iteration, the modularity

matrix is mixed randomly and 30 runs of the method were made. All measures computed from the Louvain method results (e.g. NMI) are average of the 30 runs.

9.3. Performance evaluation

Now that clustering procedures have been run and have generated clustering partitions, the quality of each partition can be calculated based on the different quality measures introduced in section 8. Note that it is impossible to compute the CCR for clustering partitions for the optimal number of clusters. Indeed, this measure needs that the clustering partition has the same number of clusters than the partition with "natural" classes. That's why comparisons involving partitions for the optimal number of clusters are only done based on NMI and ARI.

For each research question, we obtain several values for the different quality indicators on several datasets. The following types of analysis are performed depending on the case case:

1. In order to summarize all the values given by all datasets into one, an hypothesis test (Friedman test) and a multiple comparison are performed, per quality indicator, to see if there is a statistically significant difference between the methods compared. See 9.3.1 for more details.

2. Per quality indicator, we generate a two dimension graph where each axis represents a method compared and measures the scores obtained in each of the 16 datasets. Figure 9.1 shows an example where each point represents a dataset and its coordinates are determined by the scores obtained by the two methods when they were applied on the dataset. A $y = x$ line (hereafter called separation line) is drawn and the scores for all datasets are plotted. So if a method has more points on its side, it is better than the other method (in figure 9.1, method A is better). Since it consists in looking dataset per dataset, this can deliver some more insights but it is time consuming. That's why, it is only used for the second research question, the most important one.

Finally, remember that, for the Louvain method, each quality indicator is an average of the 30 runs made.

9.3.1. Friedman test and multiple comparison

The Friedman test [17, 18] is an equivalent of the repeated-measures ANOVA but it does not take a parameter as input. In [18], Friedman showed by experimentations that the ANOVA and

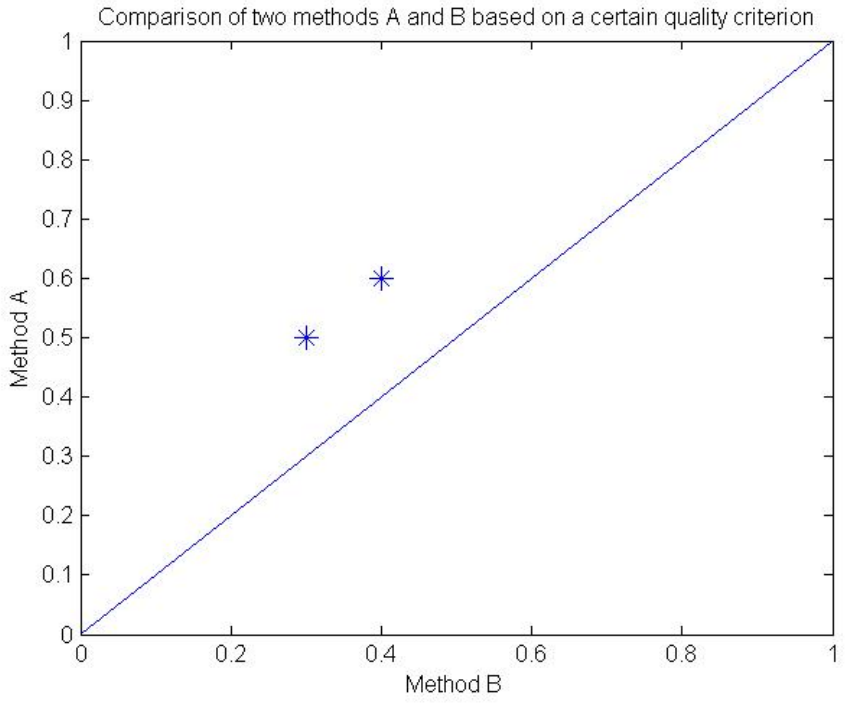


Figure 9.1.: Two-dimension graph for pairwise comparison of methods over all datasets

his test mostly agree in their conclusions. How does the Friedman test work? For each dataset, the test orders all classifiers, in our case a classifier is the score in a certain quality indicator of a partition (thus of a clustering method or kernel). Then, better is the order of a classifier, more points it gets. In other words, when a classifier is ranked 1st out of 4, it gets 4 points so that the higher number of points, the better. After, it computes the average number of points (average rank hereafter) of each classifier over all datasets.

An hypothesis test at a 95% confidence level ($\alpha = 0,05$) is performed where the null hypothesis is that all the methods are equivalent thus their average ranks equal. The Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \tag{9.1}$$

is distributed according to χ_F^2 with $k-1$ degrees of freedom, when N and k are big enough. $R_j$is the average rank of a method j from the k methods compared.

Then, if the Friedman test rejects the null hypothesis and more than 2 classifiers are compared, we are interested to find which pair of classifiers is significantly different and causes this rejection. To do that, we use a Nemenyi test in a multiple comparison, a post-hoc test used in the case where all methods are compared to each other and not against a fixed method (called a control

classifier). This test computes the following critical difference where k is the number of methods compared in total and q_αa critical value based on the Studentized range statistic divided by $\sqrt{2}$:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{9.2}$$

Any difference in rank between two classifiers that is greater than this threshold implies that these classifiers are significantly different from each other. Note that, with this type of tests, more techniques are compared, harder it is to draw conclusions from the test [10].

9.3.1.1. Post-hoc test with a control classifier

In the experiments, we also did multiple comparisons against a control classifier because the power of this test was shown to be much greater. In other words, it consists in comparing one of the methods to all the others instead of all methods to each other. It should be done when we test whether a newly proposed method is better than the existing ones [10].

Among the different tests that allow us to do this, we use the Bonferroni-Dunn test [13]. Basically, it controls the family-wise error rate by dividing α by the number of comparisons made, $k-1$ and then compute an hypothesis test statistic whose p-value is then compared to the adjusted α. However, in practice, we use an equivalent and easier to implement way of doing this test that consists in using the same critical distance and α value than for the Nemenyi test but using the critical values from table 9.4 [10].

#classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.241	2.394	2.498	2.576	2.638	2.690	2.724	2.773
$q_{0.10}$	1.645	1.960	2.128	2.241	2.326	2.394	2.450	2.498	2.539

Table 9.4.: Critical values for Bonferroni-Dunn test; the number of classifiers includes the control classifier. Source: [10]

10. Results and discussions

This chapter is meant to present, discuss and compare the results of the experiments made on the different clustering algorithms and kernels. Table 10.1 indicates the acronyms used in this thesis for the different kernels and the corresponding references to the theory so that the reader can quickly find the theoretical background. Appendices B to F include the detailed NMI, ARI and CCR scores obtained by all clustering algorithms and kernels involved in the subsequent analysis.

Full name of the kernel	Acronym	Reference to the theory
Sigmoid commute time	SCT	Equations 4.3 and 4.5
Sigmoid corrected commute time	SCCT	Equations 4.4 and 4.5
Free energy	FE	Section 4.1.2
Positive semi definite free energy	PSDFE	Section 4.1.2
Randomized shortest path	RSP	Section 4.1.2
Logarithmic forest	LF or log. forest	Section 4.1.2

Table 10.1.: Overview of the kernels: their acronyms and references to the theory

10.1. First research question

As a reminder, the first research question is: "which kernels give the best results when used in a Ward's hierarchical clustering algorithm?" and is divided in several sub-questions (see the introduction).

10.1.1. Sigmoid commute time vs. free energy kernel

First, the sigmoid commute time and the free energy kernels are compared based on their respective partitions for the "natural" number of clusters. A Friedman test on the NMI scores of the partitions shows that the two kernels cannot be differentiated, even if the SCT kernel obtains a better rank. On the visual output of the tests illustrated in figure 10.1, the x-axis represents

the mean rank of each kernel where 2 is the highest rank (i.e. the best). On the picture, the two lines are overlapping horizontally, which leads us to the conclusion that no one is significantly better than the other in terms of mean rank for the NMI. In other words, the difference between the mean ranks of the two kernels computed from the NMI scores is not significant. In this case, we even see that the two lines are strongly overlapping.

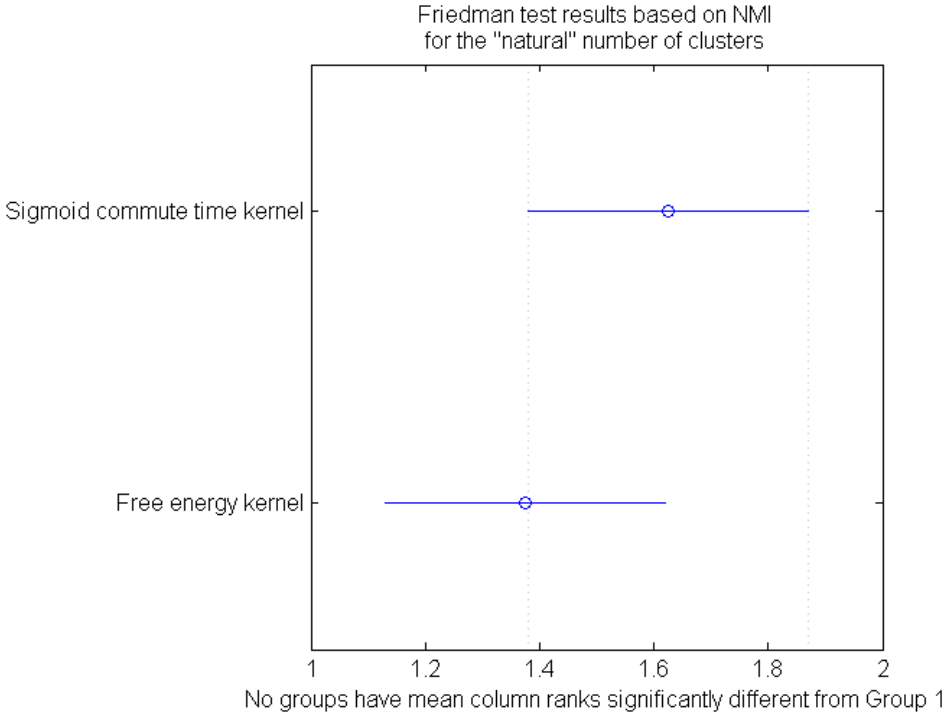


Figure 10.1.: Results of the Friedman test on the NMI scores of the sigmoid commute time and the free energy kernel ("natural" number of clusters). Rank on the x-axis ranges from 1 to 2 (the best).

A Friedman test on the ARI scores gives the same mean rank for the two kernels so they can even less be differentiated (the two lines are fully overlapping). The test on the correct classification rate produces the same output than the ARI.

The same comparison based on partitions for the optimal number of clusters supports this conclusion. The Friedman test for the ARI outputs the same mean ranks for the two kernels while in the case of the NMI, the SCT kernel has again a better rank without being significant because the two lines overlaps (see figure 10.2).

All in all, it seems that the sigmoid commute time kernel and the free energy kernel are not different in terms of quality, even if the former obtains non significantly better ranks when we take NMI as quality measure. As with any hypothesis test, no significant difference does not

mean that they are automatically equivalent. It just means that the present data do not allow us to conclude that they are different.

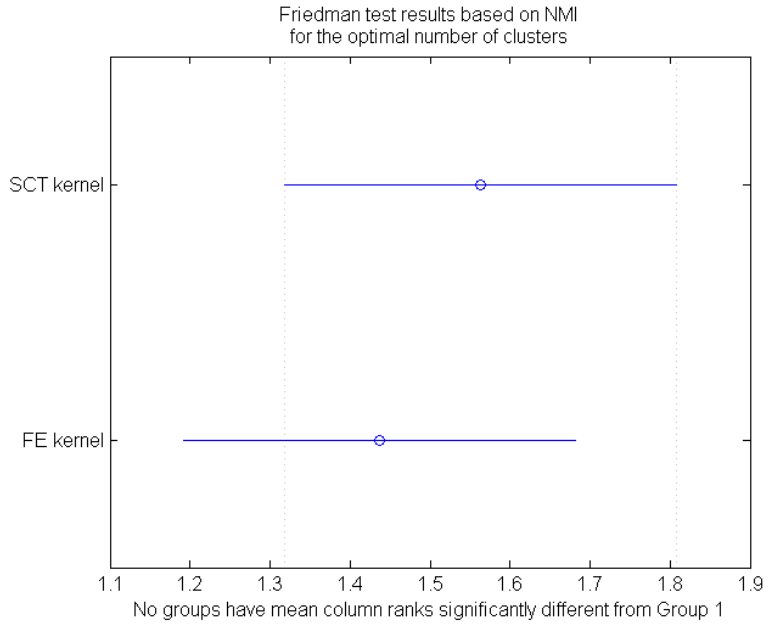
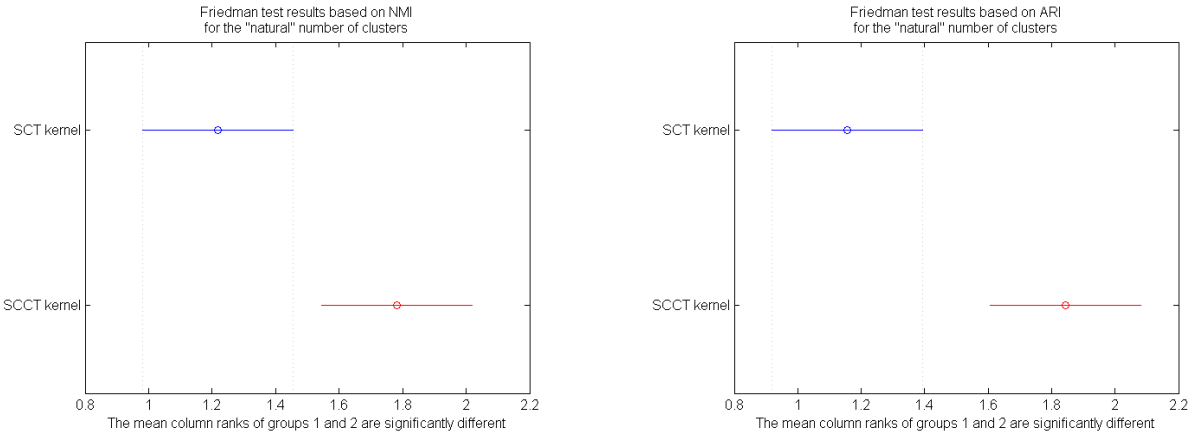


Figure 10.2.: Results of the Friedman test on the NMI scores of the sigmoid commute time and the free energy kernels (optimal number of clusters). Rank on the x-axis ranges from 1 to 2 (the best).

10.1.2. Sigmoid commute time vs. sigmoid corrected commute time kernel

The comparison based on NMI scores of the "natural" number of clusters partitions with a Friedman test shows that the SCCT kernel has a better rank than the SCT kernel. Further, the visual output shows that this difference is statistically significant because the two lines clearly does not overlap as shown in figure 10.3a. The p-value associated with the hypothesis test equals 0,02, which means that the superiority of the corrected version is significant up to a 98% confidence level. The confidence interval (at a 95% confidence level) on the difference in rank between the two kernels ranges from 0,088 to 1,037, with 0,5625 as a mean. In other words, we are 95% sure that the mean rank of the SCCT is superior by values within this range to the mean rank of the SCT kernel.

Both the adjusted rand index and the correct classification rate support this conclusion, with an even stronger p-value of 0,0045. Again, these two quality indicators lead to the same results, see figure 10.3b and Appendix G, respectively.



(a) Test on the NMI scores

(b) Test on the ARI scores

Figure 10.3.: Result of the Friedman test comparing the sigmoid commute time and the sigmoid corrected commute time kernels ("natural" number of clusters). Rank on the x-axis ranges from 1 to 2 (the best).

The same test on the clustering partitions corresponding to the optimal number of clusters gives sightly different results: the two kernels slightly overlap and this is true for both NMI and ARI. Figure 10.4 shows the visual output for the NMI and the result given by the ARI is identical (see Appendix H). The p-value associated with the tests on the NMI and ARI scores is 0,07 in both cases.

All in all, the sigmoid corrected commute time was always superior in the tests, in terms of rank, to the commute time kernel. Based on the partitions for the "natural" number of clusters, this superiority was significant up to a 98% and 99,5% confidence level based on NMI and ARI/CCR, respectively. However, it was not significant for the optimal number of clusters partitions because the p-value was only 0,07. Anyway, we can conclude that the correction of the commute time distance generated very good results compared to the original version.

10.1.3. Free energy vs. positive semi definite free energy kernel

The first experiment that we made as a pre-process stage was to verify that the free energy kernel matrix after being forced to be positive semi definite does not change too much compared to the original free energy kernel matrix. This is verified on several datasets, the matrix does not not change dramatically.

Then, as for the previous sub-questions, we compare the two kernels based on a Friedman test on the different quality indicators. For the "natural" number of clusters, the tests on all three

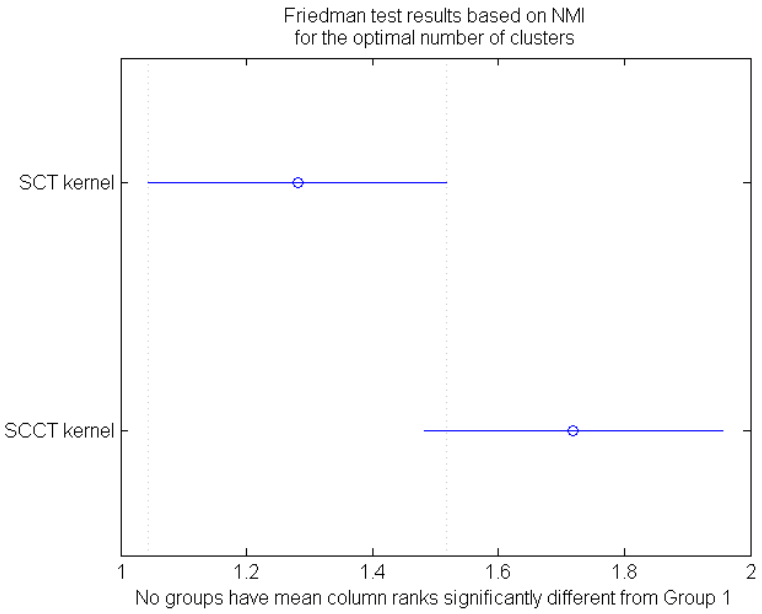


Figure 10.4.: Result of the Friedman test on the NMI scores of the sigmoid commute time and the sigmoid corrected commute time kernels (optimal number of clusters). Rank on the x-axis ranges from 1 to 2 (the best).

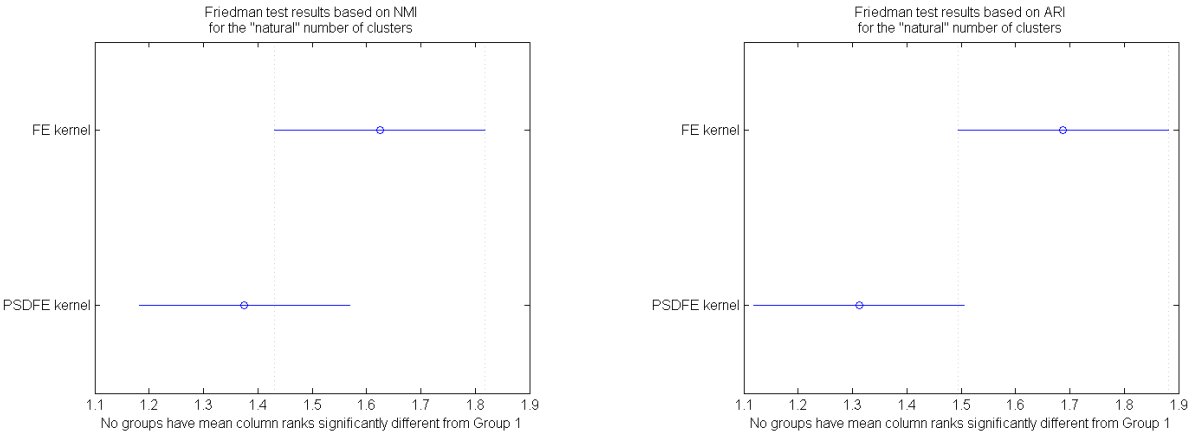
indicators assign a higher rank to the free energy kernel but the difference is not significant at a 95% confidence level. Figure 10.5a illustrates the test on the NMI scores. It should be highlighted that for the ARI (figure 10.5b) and CCR, the two kernels only slightly overlap at this confidence level. Indeed, the original free energy kernel is significantly better with a 0,0578 p-value (i.e. at a 94,22% confidence level).

For the optimal number of clusters, the tests on the scores obtained in the two indicators give a higher rank to the original free energy kernel. However, the difference in rank is again insignificant at 95% of confidence (similarly to the test on the NMI scores of the "natural" number of clusters partitions).

All in all, we can easily conclude that the positive semi definite version of the free energy kernel does clearly not generate better results than the original version, which answers our sub-question.

10.1.4. Randomized shortest path and logarithmic forest kernel

As a reminder, this sub-question is: "do the randomized shortest path and logarithmic forest kernels produce better partitions than the previously analysed kernels?". Therefore, we compare here the RSP kernel and the logarithmic forest kernel together with the SCT, SCCT and free energy kernels.



(a) Test based on NMI scores

(b) Test based on ARI scores

Figure 10.5.: Result of the Friedman test comparing the free energy and positive semi definite free energy kernels ("natural" number of clusters). Rank on the x-axis ranges from 1 to 2 (the best).

10.1.4.1. For the "natural" number of clusters

The Friedman test on NMI scores shows that there is a significant difference among this set of kernels. As a consequence, a multiple comparison with a Nemenyi test is done to compare all the possible pairs of kernels in this set. The result illustrated in figure 10.6 suggests that the RSP (as well as the FE) kernel is significantly inferior to the sigmoid corrected commute time kernel. Furthermore, we clearly see that the LF kernel is inferior to the sigmoid corrected commute time but not significantly because the two corresponding lines slightly overlap. As a consequence, we decided to perform a multiple comparison against a control classifier (procedure described in section 9.3.1.1) - in this case the LF and RSP kernels are this classifier, each one at a time - to check whether this classifier is not significantly different from one of the non control kernels. Unfortunately, this test did not generate better results as these were very similar.

For the ARI scores, the Friedman test also signals a significant difference among the kernels. As for the NMI scores, the following Nemenyi test showed that this was again because the RSP kernel is significantly inferior to the SCCT. Further, the LF kernel is again inferior - without significance - to the SCCT, the two lines overlapping a bit more than for the NMI scores. Figure 10.7 represents the multiple comparison results for the ARI scores.

Again, we tried two multiple comparisons against a control classifier, one with the RSP and one with the LF kernel as control classifier but the conclusions were the same.

The very same analysis on CCR scores leads us to the same conclusion because the output of the tests is very similar to one of the tests on ARI scores.

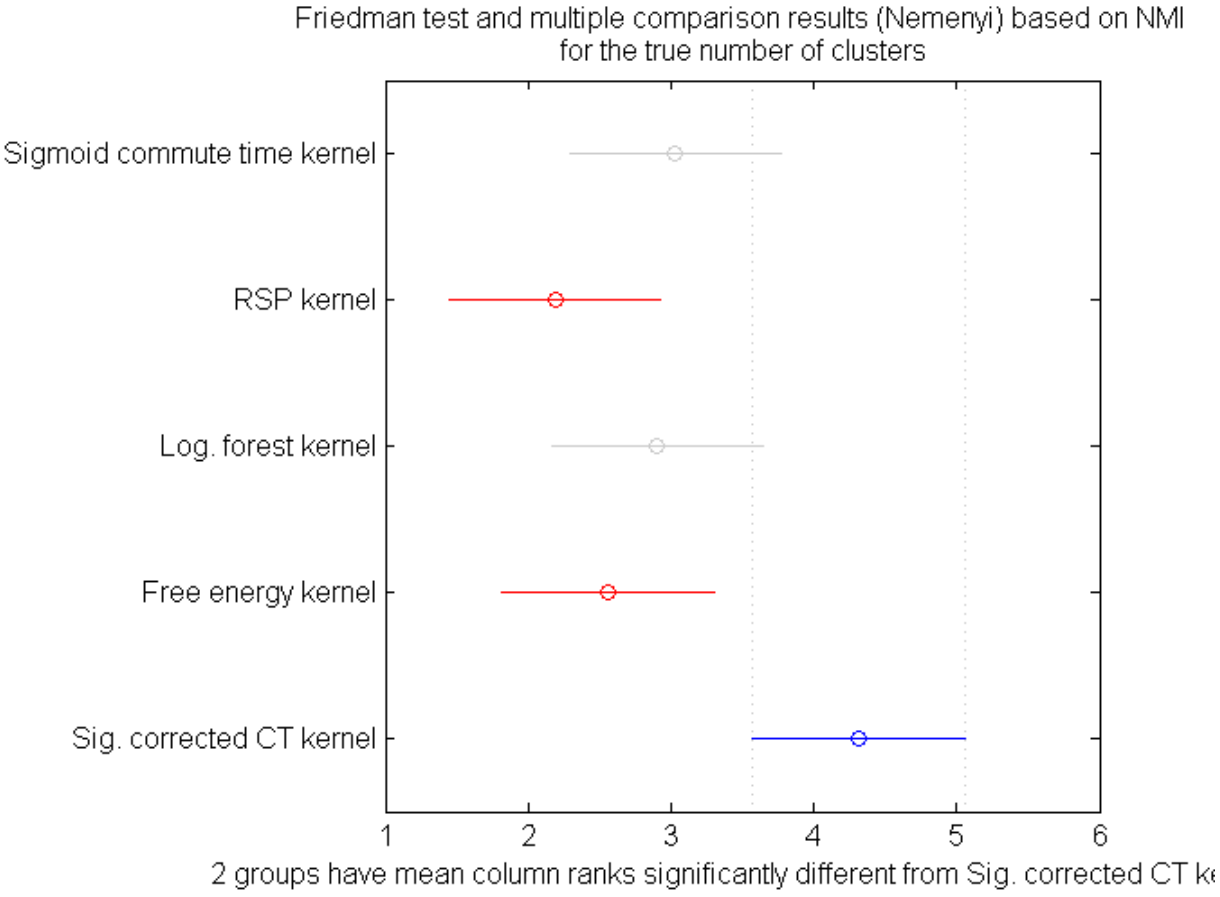


Figure 10.6.: Results of the Friedman test and post-hoc Nemenyi test on the NMI scores of the SCT, RSP, LF, FE and SCCT kernels ("natural" number of clusters). Rank on the x-axis ranges from 1 to 5 (the best).

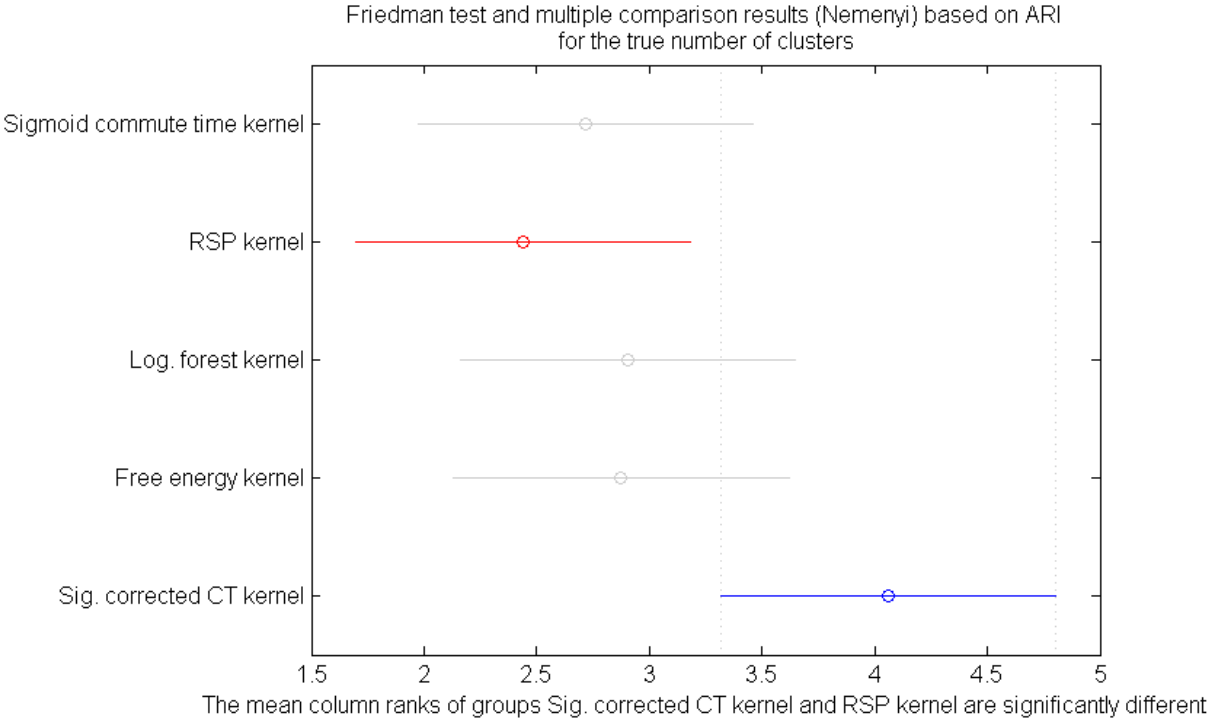
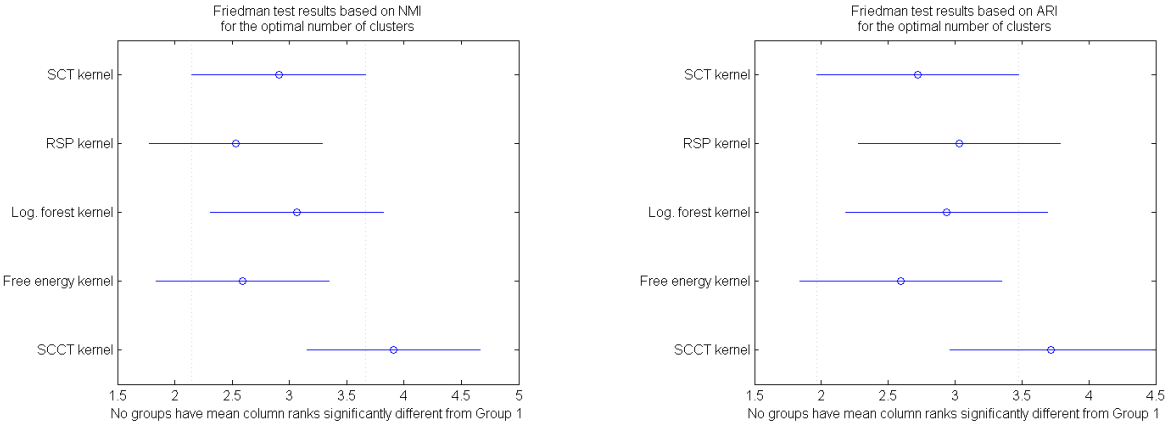


Figure 10.7.: Results of the Friedman test and post-hoc Nemenyi test (multiple comparison) on the ARI scores of SCT, RSP, LF, FE and SCCT kernels ("natural" number of clusters). Rank on the x-axis ranges from 1 to 5 (the best).

10.1.4.2. For the optimal number of clusters

When we compare the partitions for the optimal number of clusters, the Friedman tests on both NMI and ARI scores indicate no significant difference among the set of kernels, as opposed to here above. In figure 10.8b, we see that for the NMI scores the sigmoid corrected commute time is again better ranked than the RSP and the LF but the difference is not significant. The results are similar for the ARI.



(a) Test on NMI scores

(b) Test on ARI scores

Figure 10.8.: Result of the Friedman test comparing the SCT, RSP, LF, FE and SCCT kernels (optimal number of clusters). Rank on the x-axis ranges from 1 to 5 (the best).

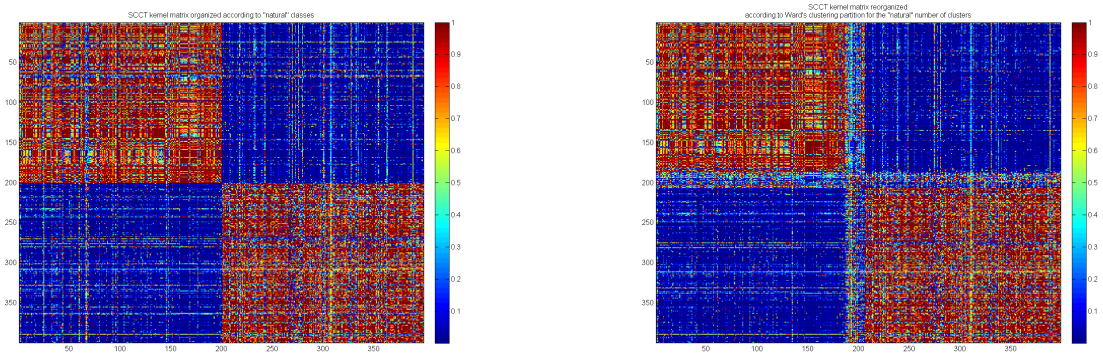
10.1.5. Conclusion

We can now wrap up the first research question that was aimed to find the best kernel. The sigmoid corrected commute time kernel clearly stands out compared to all kernels as the previous section showed. In our tests, it is consistently ranked higher than the others but in most cases without significance according to the hypothesis test. Still, the difference between this kernel and the RSP and SCT is in some cases significant.

Further, making the free energy kernel a positive semi definite matrix (thus a valid kernel) did not improve the results significantly, they tended to be even worse. Finally, the quality differential between the other kernels is much less obvious so no conclusions can be drawn about them.

To illustrate the quality of the SCCT kernel, we provide two heat maps for the newsgroup_2cl_3 dataset for which it generated good results. On the left (figure 10.9a), the heat map is the kernel matrix arranged according to the "natural" classes and on the right (figure 10.9b) the same

matrix but arranged according to the clusters of the partition for the optimal number of clusters found after running the Ward's clustering algorithm and the L method.



(a) Matrix arranged according to the "natural" classes

(b) Matrix arranged according to the clusters found by the Ward's clustering + L method (optimal number of clusters partition)

Figure 10.9.: Heat maps of the SCCT kernel matrix for the newsgroup_2cl_3 dataset. Blue (red) colours represent low (high) values of similarity.

10.2. Second research question

The second research question consists to compare the Ward's hierarchical clustering algorithm with the Louvain method (see sections 5.1 and 5.2.2 for the theory). The first sub-question relates to how well the Louvain method vs. the L method (on a Ward's hierarchical clustering partition) approach the "natural" number of clusters. The second part compares Ward's clustering partitions (for the optimal number of clusters as found by the L method) and Louvain partitions based on two quality indicators, the NMI and ARI.

10.2.1. L method vs. Louvain method and the "natural" number of clusters

For this question, the output of interest is the optimal number of clusters found by the L method on the Ward's clustering hierarchy and by the Louvain method. Appendix I and J include the details about those numbers regarding each individual dataset, when using the total within-cluster inertia and the Ward's criterion as the evaluation metric, respectively. Based on those data, was computed the absolute error on each dataset, which is simply the difference between the optimal number found and the "natural" number of clusters/classes. Then, we calculated the average absolute error obtained by each method over all datasets, which is the criterion used

to answer this sub-question. Table 10.2 presents this average for the total within-cluster inertia, which is the evaluation metric with which conclusions will be drawn (see the reasons in section 6.2).

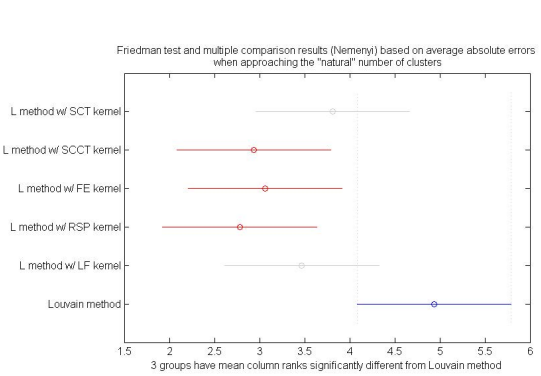
Method	L method on Ward's clustering						Louvain method
Average absolute error	**1,95**						**2,88**
Kernel	SCT	SCCT	FE	PSDFE	RSP	Log. forest	
Average absolute error per kernel	**2,31**	**1,75**	**1,87**	**1,94**	**1,75**	**2,06**	

Table 10.2.: Average absolute error with total within-cluster inertia as evaluation metric. The lower, the better.

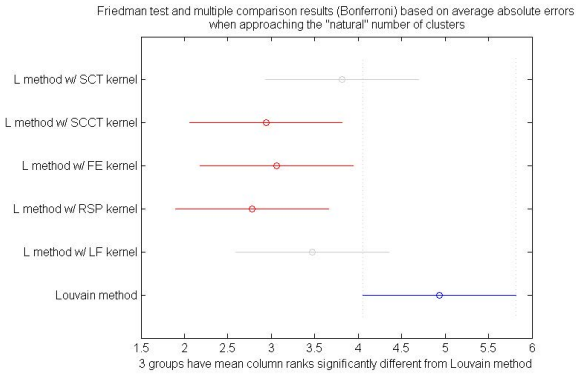
Results in the table show that the L method - whatever is the kernel as basis for the Ward's clustering algorithm - is better than the Louvain method to approach the "natural" number of clusters because the average absolute error is lower. The average of this metric over all kernels is 1,95 against 2,88 for the Louvain method. In addition, we can observe that the different kernels generate relatively different outcomes with the L method ranging from 1,75 of absolute error up to 2,31. The sigmoid commute time is clearly the worst of the kernels but is still much better, while the Louvain method and the randomized shortest path and the sigmoid corrected commute time are the best.

We decided to go further than just comparing the averages and performed an hypothesis test as for the other questions. The Friedman test confirms that there is a significant difference among the classifiers (in this case the average absolute error of each kernel and of the Louvain method), with a p-value of 0,0033. Then, we performed both Nemenyi and Bonferroni-Dunn tests in multiple comparisons. Even if the latter is supposed to give better results when the question is a comparison against a certain classifier (see section 9.3.1.1) - in this case the Louvain method - the two tests give similar results here. They show (see figure 10.10a and 10.10b) that the Louvain method is significantly inferior in terms of approaching the "natural" number of clusters to the L method when the SCCT, FE or RSP kernel is used in the underlying Ward's clustering. It is also inferior in rank to the two remaining kernels (SCT and LF) but not significantly.

Even if it made more sense in theory to use the total within-cluster inertia as evaluation metric in the L method (see section 6), in practice using the incremental inertia brought by the pair being merged (i.e. Ward's criterion) in the L method produces better results. For most of the kernels studied, using the latter decreases their average absolute error except for the sigmoid corrected commute time kernel where it increased - yet only slightly - the error. Table 10.3 presents the



(a) Nemenyi test, all methods compared to each other



(b) Bonferroni-Dunn test, comparison against a control classifier (Louvain method)

Figure 10.10.: Results of multiple comparisons following the Friedman test on the average absolute errors of the L method vs. Louvain method. Rank on the x-axis ranges from 1 to 6 (the worst).

average absolute errors and the increase or decrease when using incremental inertia instead of total inertia as above.

Kernel	SCT	SCCT	FE	PSDFE	RSP	Log. forest	Average
Average absolute error	**1,56**	**1,62**	**1,81**	**1,69**	**1,69**	**1,53**	**1,65**
Increase (decrease) in average absolute error	**-0,75**	**0,06**	**-0,18**	**-0,25**	**-0,22**	**-0,53**	**-0,31**

Table 10.3.: Average absolute error when using the Ward's criterion and difference compared to using total inertia. Lower is the average absolute error, the better.

This comparison is made over all datasets including the ones having 2 "natural" clusters. For the same reasons outlined in section 6, one may argue that this comparison is flawed because incremental inertia cannot identify 2 as optimal number of clusters. Therefore, we make the same comparison but excluding datasets with 2 "natural" clusters (i.e. 3 of the newsgroup datasets and Zachary). Results are shown in table 10.4.

Table 10.4 surprisingly indicates that using one of these two evaluation metrics does not matter so much because, at average, the corresponding average absolute errors are very close. Indeed, we would have rather expected the average absolute error of incremental inertia to decrease because it is now able to detect the same range of number of clusters than total inertia. For some kernels,

Kernel	SCT	SCCT	FE	PSDFE	RSP	Log. forest	Average
Total within-cluster inertia - average absolute error	**2,08**	**1,92**	**1,58**	**1,58**	**1,58**	**1,83**	**1,76**
Incremental inertia - average absolute error	**1,66**	**1,83**	**1,92**	**1,75**	**1,75**	**1,5**	**1,74**
Difference	**-0,42**	**-0,08**	**0,33**	**0,17**	**0,17**	**-0,33**	**-0,03**

Table 10.4.: Comparison of the average absolute errors when using incremental inertia and total inertia as evaluation metric for the L method, excluding datasets with 2 "natural" clusters. Lower is the average absolute error, the better.

incremental inertia is better similarly to the previous results of table 10.3. However, for others like the free energy kernel, total inertia is now giving better results than incremental inertia. It might be interesting to push the analysis further but since it is not a central part of this research, we consider total within-cluster inertia as good enough. In other words, it remains the evaluation metric used to answer the second research question.

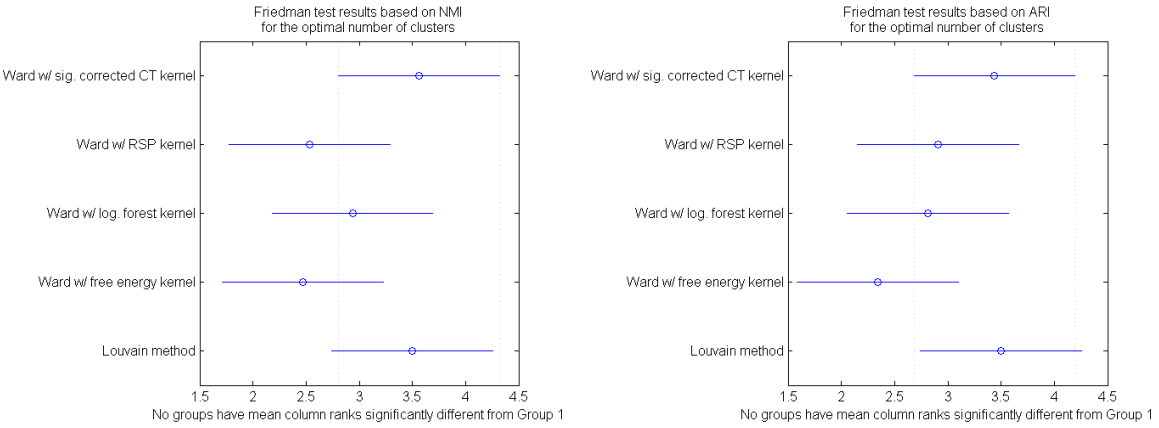
10.2.2. Ward's hierarchical clustering vs. Louvain method

We now compare the clustering partitions of the Louvain method and Ward's hierarchical clustering for the optimal number of clusters. This comparison is done based on two quality indicators, namely NMI and ARI, because the third (CCR) cannot be computed based on a partition having a different number of clusters than the "natural" number. Based on the results of the previous research questions, we decided to use only the sigmoid corrected commute time, the free energy, the logarithmic forest and the randomized shortest path kernels in Ward's clustering.

10.2.2.1. Friedman test

The Friedman test on NMI scores (figure 10.11a) indicates that the Louvain method and the Ward's clustering are not significantly different, whatever kernel is used. The same test on ARI scores (figure 10.11b) supports this conclusion. Therefore, a multiple comparison is not relevant

here since all classifiers together are not different from each other. The visual representations are provided to show the mean rank of each method. We see that the Louvain method has a better rank compared to all kernels used in Ward's clustering, with the rank of the SCCT kernel being very close.



(a) Test based on NMI scores

(b) Test based on ARI scores

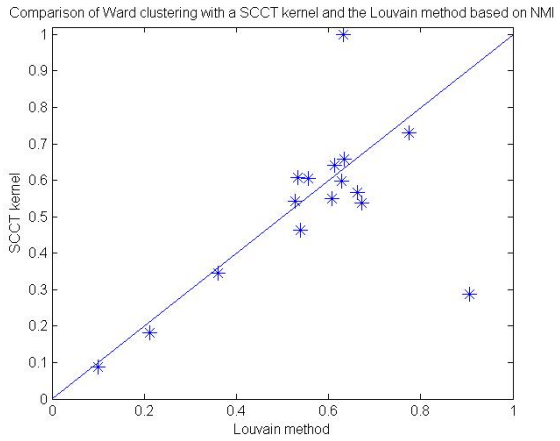
Figure 10.11.: Results of the Friedman tests on Ward's clustering partitions vs. the Louvain method. Rank on the x-axis ranges from 1 to 5 (the best).

10.2.2.2. Per dataset comparison

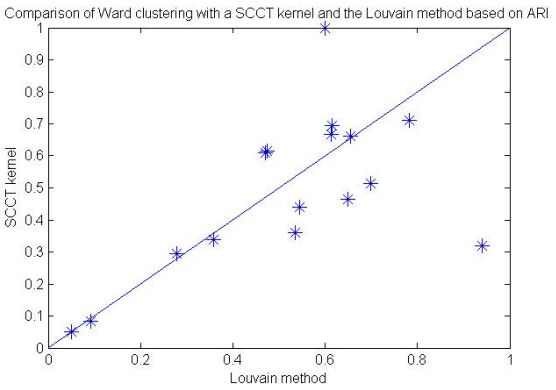
Since the Friedman test over all datasets does not distinguish the methods, let us use the two-dimension graphs introduced in section 9.3 to compare them in more details, per dataset. We thus generated 6 graphs, each one comparing the Louvain method with one kernel of the Ward's clustering based on one of the two indicators.

SCCT kernel vs. Louvain method: Looking at the position of the NMI scores of the datasets in figure 10.12a, we see that the majority is located next to the separation line except two points that are far but each one is on a different side. These two outliers are datasets for which one method gives very poor results and the other very good results. However, the Louvain method has more plots on his side (10 vs. 6) so we may say that it is slightly better. The same analysis on ARI scores (figure 10.12b) gives that plots are a bit farther from the separation line and that there is, similarly, two outliers as well. However, this time, there is almost as many points on each side. All in all, it is very difficult to differentiate the two methods here.

Free energy kernel vs. Louvain method: Again, here, the Louvain method has more plots on its side for both NMI (figure 10.13a) and ARI (figure 10.13b), even when excluding the



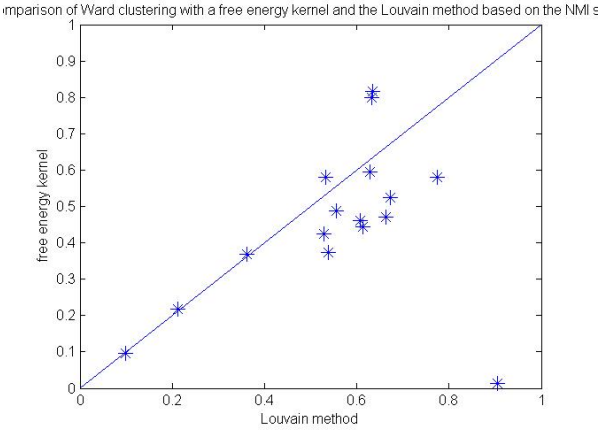
(a) Per dataset comparison based on NMI



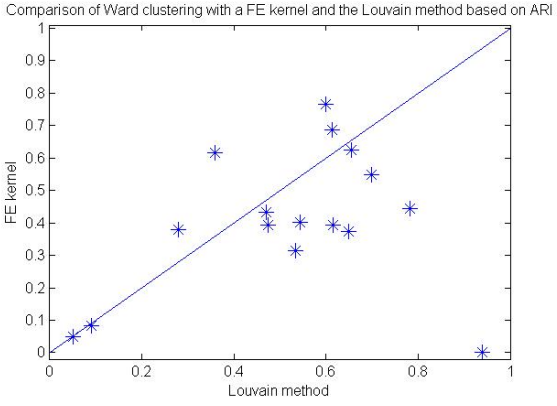
(b) Per dataset comparison based on ARI

Figure 10.12.: Per dataset comparison between SCCT kernel in Ward's clustering (y-axis) and Louvain method (x-axis)

points almost on the separation line. The two methods have their points at the same distance from the separation line (more or less) for the ARI while for the NMI the Louvain method plots are a bit farther (i.e. it is better). Further, it should be outlined in this case that the Louvain method has a point located very far from the separation line. For the corresponding dataset, results are very poor with Ward's clustering but very good with the Louvain method. All in all, it is again difficult to draw conclusions but the Louvain method is again slightly better.



(a) Per dataset comparison based on NMI

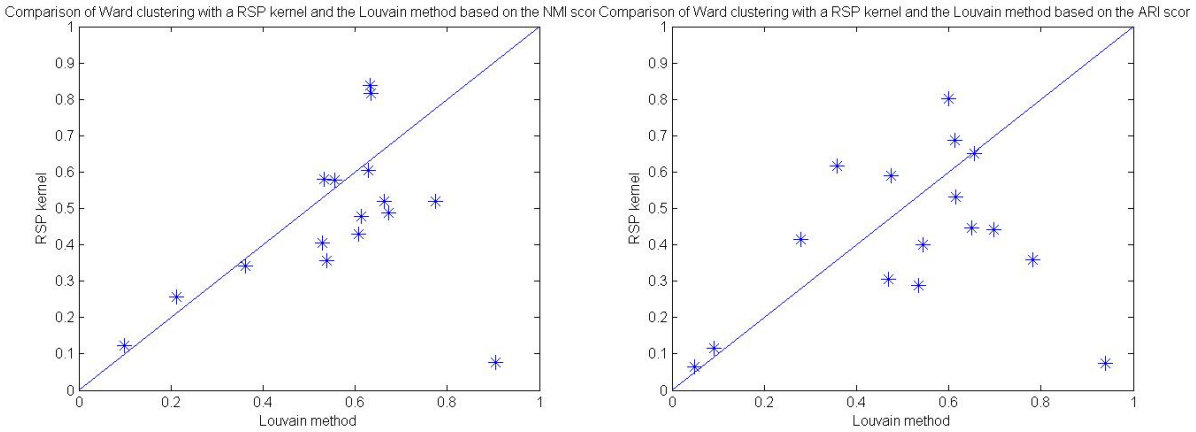


(b) Per dataset comparison based on ARI

Figure 10.13.: Per dataset comparison between FE kernel in Ward's clustering (y-axis) and Louvain method (x-axis)

RSP kernel vs. Louvain method: For the two quality measures, the Louvain method obtains 9

and 10 datasets on its side and Ward's clustering the remaining 7 and 6 datasets. Further, the Louvain method has again one outlier but not the Ward's clustering. For the NMI (figure 10.14a), most of the datasets for which the latter is better are close to the separation line while the plots on the side of the former method are a bit more distant from this line, at average. Regarding ARI scores (figure 10.14b), the two methods have their points at the same distance from the line more or less. Altogether, we can conclude that the Louvain method is slightly better than its opponent, the difference being a bit more important with this kernel than the others but still not substantial (which is in line with the results of the Friedman test).



(a) Per dataset comparison based on NMI

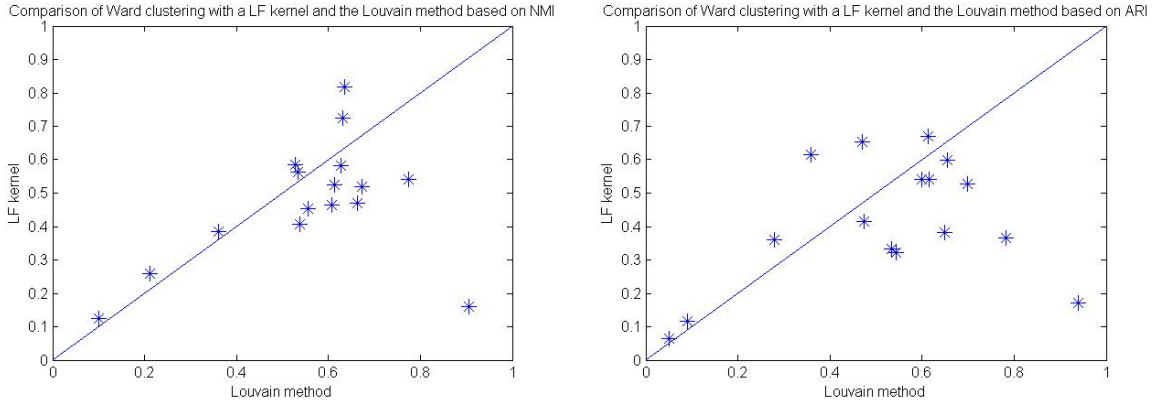
(b) Per dataset comparison based on ARI

Figure 10.14.: Per dataset comparison between RSP kernel in Ward's clustering (y-axis) and Louvain method (x-axis)

LF kernel vs Louvain method: the graphs for the two quality measures (figures 10.15a and 10.15b) lead us to the same conclusion than for the other kernels, for instance it is very similar to the ones of the RSP kernel. Again, most points (9 and 10 out of 16) are on the Louvain method side that has again one outlier. In addition, points on the Louvain method side are at average located farther from the separation line than points on the LF kernel side are.

10.2.3. Conclusion

All in all, mixed results were obtained for this second research question. On one hand, the Friedman test indicated that the quality of the partitions produced by the Ward's clustering and by the Louvain method are not significantly different, based on NMI and ARI. This was



(a) Per dataset comparison based on NMI

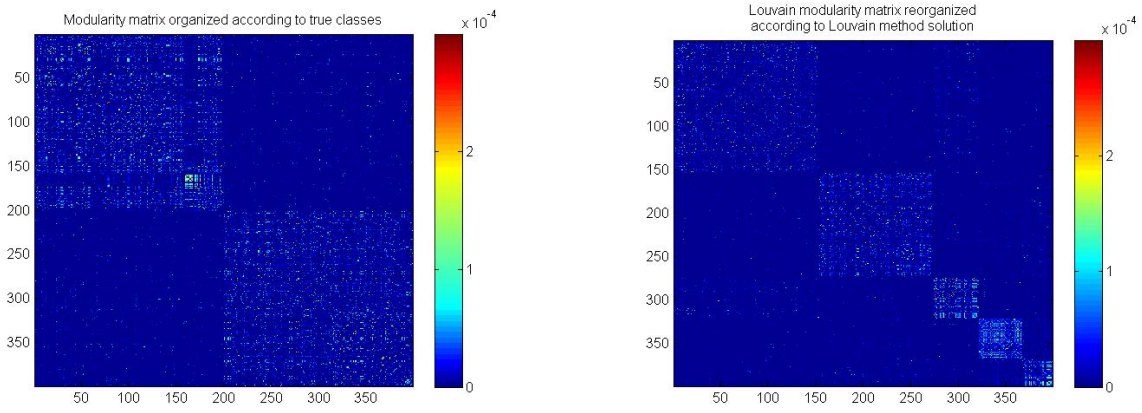
(b) Per dataset comparison based on ARI

Figure 10.15.: Per dataset comparison between LF kernel in Ward's clustering (y-axis) and Louvain method (x-axis)

confirmed when we looked dataset per dataset even if we could see the Louvain method tended to have a slight advantage.

On the other hand, when we compared them on the basis of the average absolute error that they get when approaching the "natural" number of clusters, it is the Ward's clustering on which is applied the L method that takes the advantage over its opponent, the Louvain method. Indeed, using the L method on a Ward's clustering hierarchy with a SCCT, RSP or FE kernel (but not the LF or SCT kernel) gives an average absolute error significantly lower than the one of the Louvain method.

As we did for the SCCT kernel above, we provide for the newsgroup_2cl_3 dataset a heat map representing the Louvain method modularity matrix. On the left (figure 10.16a), nodes in the matrix are grouped according to their "natural" classes whereas, on the right (figure 10.16b), the nodes are ordered according to the optimal partition determined by the Louvain method.



(a) Matrix arranged according to the 3 "natural" classes

(b) Matrix arranged according to the 5 clusters found by the Louvain method.

Figure 10.16.: Heat maps of the Louvain method modularity matrix for the newsgroup_2cl_3 dataset. Blue (red) colours represent low (high) values of modularity.

11. Managerial implications

Clustering have several applications in business and management. First, we describe its use in marketing followed by its application in social networks (thus for advertisers leveraging those networks) as it is the focus of this thesis. Finally, we present the case of Real Impact Analytics, a start-up that uses the Louvain method as well as other algorithms to analyse mobile phone networks in Africa for various purposes, including societal ones.

11.1. Clustering and marketing

Market segmentation is among the most fundamental concepts in marketing [11]. Market segments are sub-markets that share commonalities in their demand and as a consequence respond similarly to the marketing mix. Therefore, under certain conditions, segmentation can be useful for marketing practitioners [54].

Traditional clustering allows marketers to segment the market based on multivariate data about customers that typically come from market surveys and test panels. This provides a more data-driven, objective approach than segmenting based on some subjectively picked criteria, without using predictive algorithms.

Further, in marketing, we easily imagine marketers who do not have or do not want to form an idea about the *number* of segments that there should be. In this case, the L method that was covered in this thesis can be of a great help to them because we saw that it was very powerful at approaching the "natural" number of clusters. Therefore, they can use it together with a Ward's hierarchical clustering in order to determine a good partition of the market, with the right number of segments, without having to specific this number.

11.2. Clustering social networks and advertising

Social networks such as Facebook have been a major innovation in the business field during the last decade, revolutionizing how close and targeted to individuals advertising can be, as Google

had done some years before.

Engaging users as much as possible is a key success factor of Facebook strategy, even more than growing their user base [15]. Obviously, higher is the number of connections that users possess, bigger is the opportunity to engage them. Therefore, suggesting friends means supporting this engagement strategy thus creating business value. For example, assume you and your friends like the same brands on Facebook. Then, if you connect with new friends with whom you have similar interests, you will create an even more extended market of people with whom you are likely to share information about products, special offers and events [32]. In this case, this is in turn valuable for Facebook and its advertisers.

Clustering algorithms can help social networks like Facebook to determine the community to which each user belongs and then recommend to them new friends from this community [50]. That's why this thesis have important implications for companies managing social networks because it will help them identify more accurately the "natural" communities.

Further, it may sound paradoxical and counter-intuitive but it has been proved by [1] (written by Facebook employees among others) that people with whom you have weaker ties (i.e. people less close to you) are the ones responsible for the propagation of novel information even if stronger ties (i.e. the ones with people in your community) are more influential. In this case, using graph clustering to identify communities of close people and then the connections that are outside of those communities is valuable to advertisers. Indeed, connecting the people with weaker ties together is likely to improve the propagation of information about advertisements, for example.

11.3. Clustering mobile networks: the case of Real Impact Analytics

Real Impact Analytics is a start-up that mines mobile networks of telecom companies, for societal good as well as for the same telecom companies in order to help them understand their customers and target their strategy. It focuses on emerging markets, mostly from Africa.

One of their projects is aimed to fight extreme poverty by stimulating financial inclusion as it was proved by several studies that poverty results from social and financial exclusion. In emerging markets, some adults do not have a bank account so telecom operators seize this opportunity by offering Mobile Financial Service (MFC) products that help to diminish financial exclusion. However, most countries have poor adoption rates of MFC and the insights from mobile networks that Real Impact Analytics draws help to better understand how to increase this adoption rate [42].

To achieve that, Real Impact Analytics determined what drives the adoption of mobile money (MM) based on a data set of transactions including phone calls, SMS and data. Technically, those data represent affinities between mobile users (i.e. nodes) in the mobile network (see section 2.2 for the theory of networks). In [42], they found that one of the key variables that determines the propensity of a non user of MM to adopt it is the number of connections in his community already using MM as well as the level of mobile activity of those connections. It is to find those communities that the start-up applies the powerful[1] Louvain method on the mobile network [43].

Knowing all that, telecoms can offer incentives to adopt MM to people who are identified as the most likely to adopt it with the technique just described. In addition, they can target customers who are the most likely to influence others to adopt and encourage the former to actually convert the latter into MM users. For example, they can offer to those influential users currency to send via MM to non users [42].

Finding communities in mobile networks is also used to determine the probability of churn of a mobile user (i.e. probability that he/she leaves his/her operator). Indeed, they showed that if a lot of people in the community of an individual have recently changed of mobile operator, he is more likely to take the same decision.

All in all, the example of Real Impact Analytics illustrates a direct and concrete impact of graph clustering in business, more especially an impact of the Louvain method studied in this thesis. The company works for the biggest telecoms operating in emerging markets such as Vodafone, Telefonica or Orange. In addition, it helps to eradicate financial exclusion and poverty.

[1] Recognized as such on large graphs by [3]

12. Conclusion

In this thesis, we used various types of analyses to reach the more informed conclusions possible. We compared the quality of clustering methods and kernels based on several external quality indicators assessing their resulting partitions. This was done for the partition corresponding to the optimal number of clusters found by the methods but also for the "natural" number of clusters/classes, when possible. Comparisons were made with hypothesis tests at a 95% confidence interval, both pairwise and against a control classifier when more than 2 methods were compared. Finally, we also performed per dataset analyses.

12.1. Findings

In the first part of the experiments, we found that, among all kernels used as a basis for Ward's hierarchical clustering, the sigmoid corrected commute time generally stands out. More specifically, when we compare all kernels together, it is the best ranked kernel and it is generally significantly superior to the randomized shortest path and sometimes to the free energy kernel. When we compared the sigmoid corrected commute time kernel to the kernel based on the non corrected version of the same distance, we found that the correction improved significantly the results. However, it should be outlined that this only happened when comparing clustering partitions for the "natural" number of clusters. With the optimal number of clusters, however, no significant difference at a 95% confidence interval was found. Still, the superiority of the corrected version is almost significant and it is the best kernel[1] in terms of how well it is able to approach the "natural" number of clusters.

Furthermore, when we compared only the free energy with the sigmoid commute kernel together, we found no significant difference. Also, we discovered that making what we call the free energy kernel[2] positive semi definite so that it is a mathematically valid kernel did not produce significantly better results. It shows that having a similarity matrix that is not positive semi definite thus a valid kernel is good enough to use in a Ward's clustering. Of course, it would still

[1] Ex aequo with the RSP kernel
[2] Mathematically it should not be called a kernel but a similarity if it is not positive semi definite

be interesting to verify this conclusion on other similarity matrices that are not positive semi definite. In addition, one could replicate the test on more and/or different datasets than what we used because it may then show a difference.

In the second part of the experiments, the comparison between the Ward's hierarchical clustering (associated with the L method to find the optimal number of clusters in the hierarchy) and the Louvain method gave mixed results. On one hand, the L method on Ward's clustering partitions was in most cases significantly better to approach the "natural" number of clusters than the Louvain method. Whatever kernel was used in the Ward's clustering, the optimal number of clusters found by the L method was closer to the "natural" number of clusters, compared to the Louvain method. However, the difference between the two methods was only significant when using the sigmoid corrected commute time, free energy or randomized shortest path kernel in the Ward's clustering.

On the other hand, when we compared their partitions based on the quality measures, the Louvain method tended to be superior to the Ward's clustering, no matter which kernel was used in the latter. Nevertheless, it is essential to highlight that we could never reject the hypothesis that the two clustering algorithms are generating the same scores. The per-dataset analysis confirmed the relatively better results of the Louvain method and showed that, for some datasets, it gives good results while the quality of the Ward's clustering is very poor.

12.2. Limitations

This research, like any, has limitations that should be taken into account when taking the results for granted. These are:

- A great part of the datasets represented social networks. It is thus not sure that the findings are applicable to other types of networks as they were under represented here.
- All the datasets that we used were small, the largest network including 6142 nodes. This matters because we did not take into account in our analysis that for large graphs some clustering algorithms are faster than others. For example, the Louvain method is recognized to be very time efficient for large graphs.
- Due to time constraints, only one type of test designed to do a multiple comparison against a control classifier was used (the Bonferroni-Dunn test).
- We used maximum 3 (external) quality measures because adding one measure means adding a complete set of analyses and more difficulties to integrate the findings across the measures.

- Only 16 datasets were used, which was due to the rarity of network datasets including node labels indicating their "natural" class.

12.3. Further work

This research revealed new potential for further work. First, it would be interesting to test several evaluation metrics in the L method and see which one is the best to approach the "natural" number of clusters. As we did here, this comparison should be done with an hypothesis test (e.g. Friedman test) at the end to see if the difference is significant. On top of that, it is worth paying attention to which ones are able to find 2 as optimal number of clusters. Doing so was not possible with the Ward's criterion and, even if it is recognized as the best performer by the original paper [46], it is still an important limitation. Unfortunately, the very same paper does not provide us with any details about the comparison of different evaluation metrics.

Second, the research should be replicated on more datasets, bigger datasets and most varied datasets (i.e. more non social networks). On one hand, more datasets could offer more insights about the methods that could not be differentiated here. On the other hand, including a more diversified panel of datasets can permit to generalize the findings so that they would be applicable to graph clustering in general. Specifically, we would be interested in confirming the good results generated by the corrected version of the commute time distance.

Finally, it would be interesting to experimentally test the Ward's hierarchical clustering on large graphs. In order to avoid an enormous dendrogram that would not be readable, the network should first be clustered with a k-means algorithm (the interested reader can look at [59]) with a relatively small k (100 for example). This would lead to a network with 100 clusters that should then be considered as nodes of a new graph on which the Ward's clustering can be applied.

12.4. Skills acquired by the author

In this small section, we present the skills that have been acquired through this thesis and the research behind it.

Programming skills: algorithms were developed or adapted, and run with Matlab. I therefore learnt to program in Matlab, including how to produce graphics.

LaTeX: this present document was edited in LaTeX (via LyX) which I had to learn as well.

Clustering: I acquired a lot of knowledge about data mining and clustering, mostly things that I did barely know beforehand. It definitely made me more interested in the field of Big Data for my upcoming career.

Quantitative skills and Excel: analysing the output data further strengthened my quantitative skills. In addition, those data from Matlab were stored in Excel where they were processed so I improve my usage of the software.

Bibliography

[1] Bakshy, E., Rosenn, I., Marlow, C., & Adamic, L. (2012). The role of social networks in information diffusion. *Proceedings of the 21st international conference on World Wide Web*, 519-528. ACM.

[2] Barnett, S. (1992). *Matrices: Methods and Applications*. Oxford University Press.

[3] Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment,* 2008(10), P10008.

[4] Bollobás, B. (1998). *Modern graph theory (Vol. 184)*. Springer Science & Business Media.

[5] Calinski, T., Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, 3, 1–27.

[6] Campello, R. J., Moulavi, D., Zimek, A., & Sander, J. (2013). A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies. *Data Mining and Knowledge Discovery*, 27(3), 344-371.

[7] Chebotarev, P. (2011). A class of graph-geodetic distances generalizing the shortest-path and the resistance distances. *Discrete Applied Mathematics*, 159(5), 295–302.

[8] Chebotarev, P., & Shamis, E. (1997). The matrix-forest theorem and measuring relations in small social groups. *Automation and Remote Control,* 58(9), 1505–1514. http://arxiv.org/pdf/math/0602070.pdf

[9] Chung, F. R. (1997). *Spectral graph theory (Vol. 92)*. American Mathematical Society.

[10] Demsar, J. (2006). Stastical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 1-30.

[11] Dolnicar, S. (2003). Using cluster analysis for market segmentation-typical misconceptions, established methodological weaknesses and some recommendations for improvement. *Australasian Journal of Market Research*, 11(2), 5-12

[12] Duda, R. O., & Hart, P. E. (1973). *Pattern recognition and scene analysis.*

[13] Dunn, J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association*, 56, 52–64.

[14] Everitt, B. S., Landau, S., Leese, M., & Stahl, D. (2011). *Hierarchical clustering. Cluster Analysis*, 5th Edition.

[15] Facebook. (2014). *Annual Report 2013.* Retrieved on the 13th July 2015 from http://investor.fb.com/annuals.cfm

[16] Françoisse, K., Kivimäki, I., Mantrach, A., Rossi, F., & Saerens, M. (2013). *A bag-of-paths framework for network data analysis.* http://arxiv.org/abs/1302.6766

[17] Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32, 675–701.

[18] Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11, 86–92.

[19] Fouss, F., Pirotte, A., Renders, J. M., Saerens, M. (2007). *Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation.* IEEE transactions on knowledge and data engineering, 19(3), 355-369.

[20] Fouss, F., Saerens, M., Shimbo, M. (2015). *Algorithms and Models for Network Data & Link Analysis.* Cambridge: Cambridge university press

[21] Girvan, M., Newman, M. (2002). *Football dataset.* Proceedings of the National Academy of Sciences USA, 99, 7821-7826.

[22] Gobel, F. and Jagers, A. (1974). Random walks on graphs. *Stochastic Processes and their Applications*, 2, 311–336.

[23] Grinstead, C. M., & Snell, J. L. (2012). *Introduction to probability.* American Mathematical Society.

[24] Han, J., Kamber, M., & Pei, J. (2011). *Data mining: concepts and techniques: concepts and techniques.* Elsevier.

[25] Hubert, L., Arabie, P. (1985). Comparing partitions. *Journal of Classification,* 2(1), 193–218.

[26] Hubert, L. J., & Levin, J. R. (1976). A general statistical framework for assessing categorical clustering in free recall. *Psychological bulletin*, 83(6), 1072.

[27] IBorg, I., & Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media.

[28] Jain, A., Dubes, R. (1988) *Algorithms for clustering data*. Englewood Cliffs, Prentice Hall

[29] Kivimäki, I., Shimbo, M., Saerens, M. (2014). Developments in the theory of randomized shortest paths with a comparison of graph node distances, *Physica A: Statistical Mechanics and its Applications*, 393, 600-616

[30] Klein, D. J., Randic, M. (1993). Resistance distance. *Journal of Mathematical Chemistry*, 12, 81–95.

[31] Lancichinetti, A., Fortunato, S. (2009). Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1), 016118.

[32] Leone, C. (2014). *How Does Facebook Suggest Friends?* Retrieved on the 13th July 2015 from: https://blog.udemy.com/how-does-facebook-suggest-friends/

[33] Luenberger, D. (1979). *Introduction to Dynamic Systems: Theory, Models, and Applications*. John Wiley & Sons.

[34] Luxburg, U. V., Radl, A., & Hein, M. (2010). Getting lost in space: Large sample analysis of the resistance distance. *Advances in Neural Information Processing Systems*, 2622-2630.

[35] Manning, C., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge: Cambridge university press.

[36] Milligan, G. W., & Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2), 159-179.

[37] Newman, M E J. (2004). *Physical Review E*, 69, 066133.

[38] Newman, M.E.J. (2006). *Modularity and community structure in networks*. Proceedings of the National Academy of Sciences (USA), 103, 8577–8582.

[39] Newman, M. E. J. (2010). *Networks: An introduction*. Oxford University Press.

[40] Newman, M. E. J., Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 026113.

[41] Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The PageRank citation ranking: bringing order to the Web*.

[42] Real Impact Analytics. (2013). *The power of social networks to drive mobile money adoption.* Retrieved on the 10th July 2015 from: https://realimpactanalytics.com/content/data-cases/poverty-alleviation-case-power-of-social-networks-201303.pdf

[43] Real Impact Analytics. (2015). *Contact by email with Gautier Krings on the 5th July 2015.*

[44] Rokach, L., & Maimon, O. (2005). Clustering methods. *Data mining and knowledge discovery handbook*, 321-352. Springer US.

[45] Saerens, M., Fouss, F., Yen, L., & Dupont, P. (2004). *The principal components analysis of a graph, and its relationships to spectral clustering.* European Conference on Machine Learning, 371-383. Springer Berlin Heidelberg.

[46] Salvador, S., Chan, P. (2004). *Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms.* 16th IEEE International Conference on Tools with Artificial Intelligence, 576-584

[47] Santos, J. M., & Embrechts, M. (2009). *On the use of the adjusted rand index as a metric for evaluating supervised classification.* Proceeding of the 19th International Conference on Artificial Neural Networks, 175-184. Springer Berlin Heidelberg.

[48] Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis.* Cambridge university press.

[49] Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond.* MIT press.

[50] Silva, N. B., Tsang, I. R., Cavalcanti, G. D., & Tsang, I. J. (2010). *A graph-based friend recommendation system using genetic algorithm.* IEEE Congress on Evolutionary Computation 2010, 1-7.

[51] Tan, P. N., Steinbach, M., & Kumar, V. (2013). *Data Mining Cluster Analysis: Basic Concepts and Algorithms.*

[52] Theodoridis, S., Koutroumbas, K. (2008). *Pattern Recognition*, Fourth Edition (4th ed.). Academic Press.

[53] Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301), 236-244.

[54] Wedel, M., & Kamakura, W. A. (2012). *Market segmentation: Conceptual and methodological foundations (Vol. 8).* Springer Science & Business Media.

[55] Yen, L., Fouss, F., Decaestecker, C., Francq, P., & Saerens, M. (2009). Graph nodes clustering with the sigmoid commute-time kernel: A comparative study. *Data & Knowledge Engineering*, 68(3), 338-361.

[56] Yen, L., Saerens, M., Mantrach, A., & Shimbo, M. (2008, August). *A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances.* Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 785-793. ACM.

[57] Yeung, K. Y., Ruzzo, W. L. (2001). Details of the adjusted Rand index and clustering algorithms. *Bioinformatics*, 17(9), 763-774.

[58] Zachary, W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33, 452-473.

[59] Zaki, M., Meira, W. (2014). *Data Mining and Analysis: Fundamental Concepts and Algorithms.* Cambridge: Cambridge university press.