Clustering methods applied to Wikipedia

Faculté des Sciences Appliquées
Année académique 2010 - 2011

Promoteur : HUGUES BERSINI
Encadrant : JOACHIM DE BEULE
Abstract

Naturally constructed graphs, as biological or social networks, have been the subject of an increasing number of studies. The complex structure of these networks fascinates as numerous natural phenomenons rest upon them (the human conscience being the best example). For the Wikipedia graph, consisting of the articles and the hyperlinks between them, the preferential attachment rule explains part of the structure, but intuition says that the themes of each article also play a fundamental role. This work focuses on small datasets extracted from the Wikipedia database. A method allowing to extract them is presented. The clustering of the graph is performed using the recently introduced regularized commute-time kernel along with the sample-space prototype-based kernel k-means. First, the effect of normalize and center the graph kernel is assessed, with results showing an optimal performance for the centered-normalized kernel with $\alpha$ above 0.95. Then, the clustering of the Wikipedia datasets, using the graph data and article content word index, are compared and discussed. Results show that, depending on the case, both informations can lead to good or poor clusterings. In particular, some categories are so strongly linked that only the word index allows to partition them. Finally, the clusters found correspond to a great extent to the Wikipedia categories.
ACKNOWLEDGEMENTS

I WOULD LIKE TO FIRST THANK JOACHIM DE BEULE, WHO SUPERVISED ME DURING THE FIRSTS MONTHS OF THE MASTER THESIS. THE MEETINGS, CONSTRUCTIVE DISCUSSIONS AND ADVICES ALLOWED ME TO IMMESURABLY IMPROVE MY WORK. HE PRACTICALLY TAUGHT ME HOW SCIENTISTS WORK AND THINK. ALSO, MY THANKS TO PROFESSOR MARCO SAERENS FOR EXPLAINING ME THE SPHERICAL SAMPLE-SPACE PROTOTYPE-BASED KERNEL K-MEANS OF WHICH I HAD DIFFICULTIES FINDING THE EQUATIONS. HE MADE THE CALCULUS IN THIRTY MINUTES, IN FRONT OF MY WIDE OPENED EYES. I ALSO THANK AMIN MANTRACH FOR TEACHING ME HOW TO USE THE CLUSTER OF THE IRIDIA LAB AND DRIES VERDEGEM FOR SOME USEFUL DISCUSSIONS. FINALLY, I’M GRATEFUL TO PROFESSOR HUGUES BERSINI, MY SUPERVISOR, WHOSE CRITICAL VIEW ON MY WORK ALLOWED ME TO CONSIDER NEW PATHS AND IMPROVEMENTS THAT I WOULD NOT HAVE THOUGHT ABOUT ALONE.
## Contents

Abstract i
Acknowledgements ii
Introduction 1

1 Clustering 2
  1.1 Unsupervised learning 2
     1.1.1 Clustering 2
     1.1.2 K-means 3
  1.2 Kernel kmeans 5
     1.2.1 Non-linear mapping 6
     1.2.2 Kernel function and kernel trick 7
     1.2.3 Mercer’s theorem 8
     1.2.4 Sample-space prototype-based kernel k-means 9
     1.2.5 Spherical kernel k-means 11
     1.2.6 Centered and normalized kernel matrix 14
  1.3 Adjusted Rand index 15

2 Community detection in graphs 18
  2.1 Overview 18
  2.2 Commute-time kernel 18
     2.2.1 Random walker 19
     2.2.2 Average commute-time 19
     2.2.3 Commute-time kernel 20
     2.2.4 Regularized commute-time kernel 21
  2.3 Spring embedder 21

3 Textmining 22
  3.1 Word index of documents 22
  3.2 Vector space model 23
## Contents

3.3 Cosine similarity ................................................. 24  
3.4 Lucene .......................................................... 24

4 Wikipedia .......................................................... 25  
4.1 Building datasets ............................................... 26

5 Experiments .......................................................... 29  
5.1 Effects of center and normalize ............................... 29  
5.1.1 Zachary’s karate club ....................................... 29  
5.1.2 Dolphin Network ............................................. 30  
5.1.3 IMDB ......................................................... 31  
5.1.4 CORA ......................................................... 31  
5.1.5 Visualization of the kernels ............................... 32  
5.1.6 Experiment settings ....................................... 33  
5.1.7 Discussion ................................................... 34  
5.2 Clustering Wikipedia .......................................... 35  
5.2.1 Cycleways .................................................... 35  
5.2.2 Chess history ............................................... 37  
5.2.3 Fashion accessories ....................................... 39

Conclusion ............................................................ 42

Bibliography .......................................................... 42

A Fashion accessories ............................................... 47
List of Figures

1.1 K-means steps. Centroids are represented as triangles. . . . . . . . . . . 5
1.2 Non linearly separable data. . . . . . . . . . . . . . . . . . . . . . . . . . 6
1.3 Example of mapping. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
1.4 Normalized and centered data. . . . . . . . . . . . . . . . . . . . . . . . . . 14

4.1 Dataset extraction. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

5.1 Zachary’s karate club. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
5.2 Dolphins network. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
5.3 Kernels of Zachary’s karate club. . . . . . . . . . . . . . . . . . . . . . . . . 32
5.4 Results of clustering with the regularized commute-time kernel . . . . 33
5.5 Cycleways graph . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
5.6 Chess history graph. Result of graph clustering. . . . . . . . . . . . . 37
5.7 Chess history graph. Bag of words clustering. . . . . . . . . . . . . . . 38
5.8 Results of the clustering of Fashion Accessories. . . . . . . . . . . . . . 41
List of Tables

5.1 Comparison of the centered-normalized regularized commute-time and the sigmoid commute-time .................. 34
5.2 5 most important articles of each cluster, Cycleways. .............. 36
5.3 10 most important articles of each cluster, Chess history. ........ 38
5.4 10 most important articles of each cluster, Chess history. ........ 39

A.1 Links clustering, part 1. ............................................. 47
A.2 Links clustering, part 2. ............................................. 48
A.3 Word index clustering, part 1. .................................... 48
A.4 Word index clustering, part 2. .................................... 49
Introduction

This master thesis follows the results of Beule et al. [2]. In his paper, he describes a model to improve the clustering of documents by using the knowledge of the Wikipedia database. De Beule linked the documents to Wikipedia articles, and used the first step of hyperlinks between articles to complement the word vector representation of the documents. Many questions arises then. Could we find a way to use the categories of Wikipedia? Is it possible to cluster the Wikipedia articles to obtain clean labels? More generally, what happens when we cluster Wikipedia?

The English Wikipedia database consist of 3.5 millions of articles, belonging to 650,000 categories\(^1\). Anyone can write an article, associate it to categories and define links to and from other articles to attach it to the rest of the database. The graph consisting of the articles and the hyperlinks between them is thus a collaboratively build network, and is what makes it interesting to study.

The subject is quite vast and attractive. The aim of this work is to try to add some contribution, as little as it may be, to this research domain by extracting small parts of the graph of articles, apply clustering on the graph and on the word index, and see what happens. To characterize and compare the clusters will then be an interesting analysis.

\(^1\)Statistics of February 2011
Chapter 1

Clustering

1.1 Unsupervised learning

Unsupervised learning is a class of machine learning algorithms, where the aim is to find patterns in raw, unlabeled, data. Unlike in supervised learning, where the algorithm is first trained on labeled data, so it can learn and adapt to the particular problem (classification), we use here general, inherent, properties of the data to find patterns and organize it (clustering).

Since the two domains are quite large, all the energy during this master thesis has been focused on unsupervised learning, and more specifically on clustering.

1.1.1 Clustering

The goal of clustering is to define clusters of elements in the dataset, such that the elements in the same cluster are similar to each other, and each cluster as a whole is distant from the others. To perform these partitions, a similarity measure needs to be defined, to be able to compute the distance between the elements of the dataset. Choosing the best similarity measure is not an easy task. The performance of each can vary depending on the problem at hand. Also, the data can present itself in various different ways, as vectors of categorical, real, integer, etc, values. The most common measure is the euclidian distance.

Two important classes of clustering can be distinguished:
Chapter 1 : Clustering

Hierarchical clustering: These kind of techniques can either be agglomerative or divisive. In agglomerative clustering, we start by assigning each element to a different class. The successive iterations of the algorithm cluster together the closest classes, until all the elements belongs to the same, main, class. The advantage of hierarchical clustering is that we get the general structure of the data, often represented as a dendogram, and we can choose the level of clustering. The divisive methods performs divisions of the classes into smaller ones instead.

Partitional clustering: The idea of partitional clustering is to divide the data immediately into a certain number of clusters. The most popular algorithm is k-means, which will be presented in the next section.

Other classes of clustering, like subspace clustering, will not be discussed here.

1.1.2 K-means

The k-means algorithm was proposed independently in various scientific fields over 50 years ago (Jain [18]). It follows that it is known under different names (nuées dynamiques, dynamic clusters method, iterated minimum-distance partition method, nearest centroid sorting). MacQueen was the first, in 1967, to name k-means its one-pass version of the algorithm, where he defined the first k elements of the dataset as the k classes, and successively assigned the next elements to the closest class, updating the centroid after each assignment (Bock [4]).

The standard k-means algorithm is considered to be a simple but efficient partitional algorithm. It divides the data into $k$ clusters, minimizing the squared distance between each element to the center of its cluster. The distance measure is a parameter of the algorithm. The objective function, using the euclidian distance, is defined as:

$$J = \sum_{k=1}^{m} \sum_{i \in C_k} \|x_i - g_k\|^2$$

(1.1)

Where:

- $m$: is the total number of clusters.
- $C_k$: is the k-th cluster.
- $x_i$: is the vector of the i-th element of the dataset.
• $g_k$: is the vector of the center of the k-th cluster.

Minimizing this function is a NP-hard problem (Jain [18]). K-means then proposes the following iterative method to find a good solution.

1. Initialize the center of each cluster (also called centroids). For example, we can arbitrarily choose some elements of the dataset to be the centroids.

2. Reassign each element to the closest centroid.

3. Recompute the center of each cluster.

4. Repeat steps 2 and 3 until the stopping criterion is satisfied.

The stopping criterion can be for example when the decrease of the objective function stabilises:

$$\left| J_{\text{new}} - J_{\text{old}} \right| < \delta_{\text{threshold}}$$ (1.2)

The algorithm always converges to a local minimum of the objective function, thus it is very sensitive to the initialization. The steps are illustrated in figure 1.1.

The two main steps, the computation of the centroids and the reassignment of the elements, are formally defined below.

**Computation of the centroids:** the center of each cluster is the mean of the sum of all the vectors belonging to the cluster (Bontempi [5]):

$$g_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$ (1.3)

Where $|C_k|$ is the size of cluster $k$.

**Reassignment:** we define the assignment vector $l$ as:

$$l_i = \arg \min_k \{ \|x_i - g_k\| \}$$ (1.4)

The i-th element of $l$ thus contains the label (cluster id) of the cluster to which the vector $i$ belongs.

This master thesis is focused on the clustering of graphs. To apply k-means to a graph, we need to define the similarity measure between the nodes (in this case, through a random walker). However, this is to be seen in the more general context of kernels and kernel functions. Before introducing the regularized commute-time kernel, we first present the use of kernels in machine learning.
Figure 1.1: K-means steps. Centroids are represented as triangles.

1.2 Kernel kmeans

One of the limitations of k-means is that it is only able to find clusters when they are linearly separable. The data shown in figure 1.2, for example, is incorrectly classified by the standard k-means algorithm since an hyperplane cannot be drawn between the two clusters.

One of the solutions to this problem is described below, and will lead to the definition of kernels.
1.2.1 Non-linear mapping

Since a linear separation of the data is not possible in the input space, the idea is to map the data into a higher dimensional space, a feature space, in a non-linear way and apply the same algorithm in this new space.

The mapping is defined as follows (Müller et al. [26]):

$$\phi : \mathbb{R}^N \rightarrow \mathcal{F}$$
$$x \mapsto \phi(x)$$

The $n$ elements $x_i$ are mapped from their N-dimensional $\mathbb{R}^N$ input space to the higher dimensional $\mathcal{F}$ feature space.

A widely known example from Schölkopf and Smola [31] is shown in figure 1.3. By applying the following mapping:

$$\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

the ellipsoidal boundary in input space becomes a hyperplane in feature space.

This allows to keep the algorithm unchanged, and use it on transformed data. However, the mapping $\phi$ must still be defined and then computed. Due to the
number of dimensions, often large, the complexity of the computation rises and can make real life problems impossible to solve in practice (see Schölkopf and Smola [31, pg. 27] for an example). Hopefully, a trick exists.

1.2.2 Kernel function and kernel trick

Before introducing the kernel trick, we notice that k-means uses the squared euclidian distance to cluster the data. In input space, the squared distance between two vectors is given by $\|u-v\|^2$. After applying a mapping $\phi$, the distance in feature space is $\|\phi(u) - \phi(v)\|^2$. This can be expressed as:

$$
\|\phi(u) - \phi(v)\|^2 = \phi^2(u) - 2\phi(u)\cdot\phi(v) + \phi^2(v)
= \phi(u)\cdot\phi(u) - 2\phi(u)\cdot\phi(v) + \phi(v)\cdot\phi(v)
$$

This is the measure of distance used by k-means to cluster the data, after the non-linear mapping. The distance depends only on scalar products. If we consider
Chapter 1 : Clustering

that the input space is $\mathbb{R}^2$ and apply the same mapping than in the example in the previous subsection:

$$
\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3
$$

$$(u_1, u_2) \mapsto (z_1, z_2, z_3) := (u_1^2, \sqrt{2}u_1u_2, u_2^2)$$

we can express the scalar product in feature space as (Hofmann [16]):

$$
\phi(u) . \phi(v) = \left( u_1^2, \sqrt{2}u_1u_2, u_2^2 \right) . \left( v_1^2, \sqrt{2}v_1v_2, v_2^2 \right) = \left( u_1v_1 + u_2v_2 \right) = (u.v)^2
$$

The scalar product between two vectors in feature space can be computed without explicitly knowing the mapping $\phi$. The value $(u.v)^2$ is defined as the kernel function:

$$
k(u, v) := \phi(u) . \phi(v)
$$

To each kernel function correspond a mapping. This kernel function depends only on vectors from the input space, which means that we do not need to define a mapping anymore. This is known as the kernel trick.

Many kernel functions can be found in the literature, from the simplest polynomial one to the classic gaussian (Souza [32] lists nearly twenty five different kernels). To be considered as kernel, a function must correspond to a mapping. All the kernels used during this project have been validated in other studies, but as a side note, the next section will give a necessary and sufficient condition for this to be true.

1.2.3 Mercer’s theorem

Considering a dataset of $n$ elements to cluster, represented by the vectors $x_1, ..., x_n$, we can build a matrix containing the scalar products between all the elements. A kernel function is used to implicitly map the elements to the feature space:

$$
K = \begin{pmatrix}
\phi(x_1).\phi(x_1) & \ldots & \phi(x_1).\phi(x_n) \\
\vdots & \ddots & \vdots \\
\phi(x_n).\phi(x_1) & \ldots & \phi(x_n).\phi(x_n)
\end{pmatrix} = \begin{pmatrix}
k(x_1, x_1) & \ldots & k(x_1, x_n) \\
\vdots & \ddots & \vdots \\
k(x_n, x_1) & \ldots & k(x_n, x_n)
\end{pmatrix}
$$
This matrix, defined as the **Kernel matrix**, thus contains the similarities between all the elements of the dataset. It completely represents the data and is all we need to perform a clustering. To be a valid kernel, \( K \) must satisfy the Mercer’s theorem (Ng [28]):

**Mercer’s theorem.** Let \( K(x, z) \) be given. Then, \( K \) is a valid (Mercer) kernel (i.e. \( \exists \phi \text{ s.t. } K(x, z) = \phi(x)^T \phi(z) \)) if and only if for any set \( \{x_i, ..., x_n\} \) (\( n < \infty \)) the kernel matrix \( K \in \mathbb{R}^{n \times n} \) is positive semi-definite.

This ensures that \( K \) is a Gram matrix (a matrix of inner products).

Kernel functions and the kernel trick are not specific to k-means. They are for example widely used in Support Vector Machines and contributed to make them popular. Kernels can be used with any algorithm based on scalar products. Papers from Müller et al. [26], Girolami [13] or the book from Schölkopf and Smola [31] give a more general view of kernel-based learning.

### 1.2.4 Sample-space prototype-based kernel k-means

Now that the kernel matrix is defined, we need to rewrite the formulas of the k-means algorithm, and replace every scalar product with the corresponding value of the kernel. For now on, to avoid overload of notations, we define \( u_i = \phi(x_i) \) as the vectors in feature space and \( z_k \) as the centroids in feature space.

One difficulty arises, which is the computation of the centroids. Since all the vectors are mapped using \( \phi \), the formula 1.3 becomes:

\[
  z_k = \frac{1}{|C_k|} \sum_{i \in C_k} u_i
\]

However, we do not know the values of \( u_i \). The problem can be tackled by keeping \( z_k \) as a sum of vectors, and using kernel values at the following step (the reassignment to the closest cluster) (Zhang and Rudnicky [36]). We will adopt another method here, from Yen et al. [34], which is more general and can be applied to other kernel clustering algorithms as well.

The key idea of sample-space prototype-based kernel clustering is to define the centroid vector in *sample space* \( h_k \):

\[
  z_k \rightarrow X^T h_k
\]


Chapter 1: Clustering

Where $X = [u_1, ..., u_n]^T$, $h_k$ has one dimension per element of the dataset. It can be seen as a kind of assignment vector. The $i$-th element of $h_k$ is $> 0$ when the vector $i$ belongs to cluster $k$, and 0 otherwise (see below for the exact expression).

Using this new centroid vector, we can rewrite the objective function (Yen et al. [34]):

$$ J = \sum_{k=1}^{m} \sum_{i \in C_k} \|u_i - z_k\|^2 $$

$$ = \sum_{k=1}^{m} \sum_{i \in C_k} (x_i - z_k)^T(x_i - z_k) $$

$$ = \sum_{k=1}^{m} \sum_{i \in C_k} (x_i^T x_i - 2x_i^T z_k + z_k^T z_k) $$

$$ = \sum_{k=1}^{m} \sum_{i \in C_k} (x_i^T x_i - 2x_i^T X^T h_k + h_k^T X^T h_k) $$

$$ J = \sum_{k=1}^{m} \sum_{i \in C_k} (e_i - h_k)^T K (e_i - h_k) $$ \hspace{1cm} (1.5)

The two steps of k-means follow.

**Reassignment:** as in formula 1.4, the vector $l$ is such that the objective function is minimized:

$$ l_i = \arg \min_k \{ (e_i - h_k)^T K (e_i - h_k) \} $$ \hspace{1cm} (1.6)

**Computation of the centroids:** this step is a little more tricky. We want to recompute the center of each cluster. That is, to minimize the objective function, while keeping the assignments fixed. In other words, to set the gradient of $J$ with respect to $h_k$ equal to zero:

$$ \frac{\delta J}{\delta h_k} = \frac{\delta}{\delta h_k} \left[ \sum_{k=1}^{m} \sum_{i \in C_k} (e_i^T K e_i - e_i^T K h_k - h_k^T K e_i + h_k^T K h_k) \right] = 0 $$
⇒ \sum_{i \in C_k} (-2e_i^T K + 2n_k h_k^T K) = 0

h_k^T K = \frac{1}{n_k} \sum_{i \in C_k} e_i^T K

K h_k = K \frac{1}{n_k} \sum_{i \in C_k} e_i

Where \( n_k \) is the number of elements belonging to cluster \( k \). The last line is a linear system of equations. One solution is given by Yen et al. [34]:

\[
\begin{align*}
  h_k &= \frac{1}{n_k} \sum_{i \in C_k} e_i \\
\end{align*}
\]

(1.7)

The i-th element of \( h_k \) is equal to \( \frac{1}{n_k} \) when the vector \( i \) belongs to cluster \( k \), and 0 otherwise.

All the formulas for kernel k-means are now defined. These can be used with any kernel. The next section presents a normalized version of kernel k-means.

### 1.2.5 Spherical kernel k-means

In standard kernel k-means, we seek the classes \( C_k \) which minimizes the sum of the squared distance from each element \( u_i \) to the centroid of their class \( z_k \):

\[
J = \sum_{k=1}^{m} \sum_{i \in C_k} \| u_i - z_k \|^2
\]

In spherical kernel k-means, we normalize each vector. This means that vectors with different norms, but pointing in the same direction, will become similar and clustered together. This will become useful when clustering word vectors of documents. After normalization, the measure of distance becomes:

\[
\| u_i - z_k \|^2 = \| u_i \| + \| z_k \| - 2 u_i z_k = 2 - 2 \cos(u_i, z_k)
\]

Only the cosine remains as a measure of similarity between each element. This will affect the formulas of kernel k-means. Instead of minimizing the objective function, we seek this time the classes which maximize the sum of the cosine between each element and the centroid of their class:
\[
J_S = \sum_{k=1}^{m} \sum_{i \in C_k} \cos(u_i, z_k) = \sum_{k=1}^{m} \sum_{i \in C_k} \frac{u_i^T z_k}{\|u_i\| \|z_k\|}
\] (1.8)

We now express the formulas for the reassignment of the elements to the closest centroid and the recomputation of the centroids\footnote{Big thanks to Prof. Marco Saerens for explaining me the calculus}. As in standard kernel k-means, we introduce the centroid in sample space:

\[z_k \rightarrow X^T h_k\]

Since \(u_i^T X^T h_k = (Xu_i)^T h_k = k_i^T h_k\), the objective function becomes:

\[
J_S = \sum_{k=1}^{m} \sum_{i \in C_k} \frac{k_i^T h_k}{\|u_i\| \|X^T h_k\|}
\]

\textbf{Reallocation:} For each element \(i\), we seek the class \(k\) that maximizes the corresponding term in the objective function:

\[l_i = \text{arg max}_k \left[ \frac{k_i^T h_k}{\|u_i\| \|X^T h_k\|} \right]\]

Since the centroids are normalized (see next paragraph), \(\|X^T h_k\| = 1\) and:

\[\Rightarrow l_i = \text{arg max}_k \left[ \frac{k_i^T h_k}{\sqrt{k_{ii}}} \right]\] (1.9)

\textbf{Computation of the centroids:} To impose the fact that the norm of the centroids \(\|z_k\| = \|z_k\|^2 = 1\), we use the Lagrange multipliers method:

\[
\Lambda = \sum_{k=1}^{m} \sum_{i \in C_k} \frac{k_i^T h_k}{\|u_i\|} + \sum_{k=1}^{m} \lambda_k (z_k \cdot z_k - 1)
\]

\[
= \sum_{k=1}^{m} \sum_{i \in C_k} \frac{k_i^T h_k}{\|u_i\|} + \sum_{k=1}^{m} \lambda_k (h_k^T X^T h_k - 1)
\]

We also add a factor \(-\frac{1}{2}\) to the second term to simplify the coming calculus.
\[ \Lambda = \sum_{k=1}^{m} \sum_{i \in C_k} \frac{k_i^T h_k}{\|u_i\|} + \frac{1}{2} \sum_{k=1}^{m} \lambda_k (1 - h_k^T K h_k) \]

The partial derivative with respect to \( h_k \) allow us to find an expression of the centroids:

\[
\sum_{i \in C_k} \frac{k_i}{\sqrt{k_{ii}}} - \lambda_k K h_k = 0 \quad (1.10)
\]

\[
\Rightarrow K h_k = \frac{1}{\lambda_k} \sum_{i \in C_k} \frac{k_i}{\sqrt{k_{ii}}}
= \frac{1}{\lambda_k} \sum_{i \in C_k} K e_i \sqrt{k_{ii}}
= K \frac{1}{\lambda_k} \sum_{i \in C_k} e_i \sqrt{k_{ii}}
\]

\[
h_k = \frac{1}{\lambda_k} \sum_{i \in C_k} \frac{e_i}{\sqrt{k_{ii}}} \quad (1.11)
\]

To find the expression of \( \lambda_k \), we multiply (1.10) by \( h_k^T \):

\[
\lambda_k h_k^T K h_k = \|z_k\|^2 = 1
\]

\[
\lambda_k = \left( \sum_{i \in C_k} \frac{k_i^T}{\sqrt{k_{ii}}} \right) h_k
\]

And then replace \( h_k \) by (1.11):}

\[
\lambda_k^2 = \sum_{i \in C_k} \frac{k_i^T}{\sqrt{k_{ii}}} \sum_{j \in C_k} \frac{e_j}{\sqrt{k_{jj}}}
= \sum_{i,j \in C_k} \frac{k_{ij}}{\sqrt{k_{ii}k_{jj}}}
\]

\[
\lambda_k = \sqrt{\sum_{i,j \in C_k} \frac{k_{ij}}{\sqrt{k_{ii}k_{jj}}}} \quad (1.12)
\]
As we can see, the term $\sqrt{k_{ii}}$ always appear at the denominator. We can then simplify the formulas by normalizing the kernel matrix beforehand.

### 1.2.6 Centered and normalized kernel matrix

![Normalized and centered data](image1)

Figure 1.4: Normalized and centered data. In black, the original data. In red, the transformed one. The unit circle is drawn in blue.

Figure 1.4 shows the effect of normalizing and center vectors in two dimensions. Centering the data before normalizing it gives a different output than normalizing it right away. Experiments presented in section 5.1.6 explore how these transformations affect the performance of the clustering. As we can see, normalizing and then center is not very meaningful. The data distribution is similar to a simple normalization, but the normalization is lost.
Chapter 1 : Clustering

The normalization of the kernel matrix is performed by the following formula:

\[ K_n = D^{-1/2} K D^{-1/2} \]  
(1.13)

Where:

\[ D = \begin{pmatrix} k_{ii} & 0 & \ldots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & k_{nn} \end{pmatrix} \]

is the diagonal of \( K \). Each value of the matrix becomes \( (K_n)_{ij} = \frac{k_{ij}}{\sqrt{k_{ii}k_{jj}}} \). In particular, the diagonal values of \( K_n \) are equal to 1. Indeed, \( (K_n)_{ii} = u_i \cdot u_i = \|u_i\|^2 = \frac{k_{ii}}{\sqrt{k_{ii}k_{ii}}} = 1 \).

Centering of the kernel matrix can be done with the centering matrix \( H \):

\[ H = I - \frac{e \cdot e^T}{n} \]

Where \( e \) is a vector of all 1’s of size \( n \). The centered kernel matrix is then obtained with:

\[ K_c = HKH \]  
(1.14)

1.3 Adjusted Rand index

Results of clustering vary, depending on the method used and the parameters. To assess the quality of a partition then becomes a central issue. We must be able to compare methods to choose the most appropriate algorithm for our class of problems, and find the best parametrization before applying them to new problems.

The measure used during this master thesis has been introduced by Rand [29] in 1971 as a way to compare two classifications.

Suppose we have a set of objects \( X_1, \ldots, X_n \) and the classifications \( U = \{u_1, \ldots, u_R\} \) and \( V = \{v_1, \ldots, v_C\} \). William Rand defines his index as:
Chapter 1 : Clustering

\[ c(U, V) = \sum_{i<j}^{n} \frac{\gamma_{ij}}{\binom{n}{2}} \]

\[ \gamma_{ij} = \begin{cases} 
1 & \text{if } X_i \text{ and } X_j \text{ are classified in the same group in } U \text{ and } V \\
1 & \text{if } X_i \text{ and } X_j \text{ are classified in different groups in both } U \text{ and } V \\
0 & \text{otherwise} 
\end{cases} \]

\( c \) ranges from 0 when there is no similarity between the classifications, to 1 when they agree. It is based on how pairs of objects are classified.

As explained by Hubert and Arabie [17], there are four kinds of classifications among \( \binom{n}{2} \):

- **a**: both objects are placed in the same group in \( U \) and \( V \).
- **b**: both objects are placed in different groups in \( U \) and \( V \).
- **c**: both objects are placed in the same group in \( U \) and in different groups in \( V \).
- **d**: both objects are placed in different groups in \( U \) and in the same group in \( V \).

\( a \) and \( b \) are considered agreements, while \( c \) and \( d \) are disagreements. The Rand index is the sum of agreements divided by the number of possible pair combinations.

Hubert and Arabie [17] further improved the index by defining the *adjusted Rand index*. It comes from the observation that when comparing random classifications, the Rand index does not take a constant value. This makes difficult the comparison with other indexes. The idea is thus to correct the index using the expected value when both classifications are random (the null model chosen). The formula used is the following:

\[
\text{Index} - \text{Expected Index} \\
\frac{\text{Maximum Index} - \text{Expected Index}}{}
\]

Consequently, Hubert and Arabie [17] brings to light the formula for the adjusted Rand index:
\[
\sum_{i,j} \binom{n_{ij}}{2} - \sum_{i} \binom{n_i}{2} \sum_{j} \binom{n_j}{2} / \binom{n}{2}
\]

\[
\frac{1}{2} \left[ \sum_{i} \binom{n_i}{2} + \sum_{j} \binom{n_j}{2} \right] - \sum_{i} \binom{n_i}{2} \sum_{j} \binom{n_j}{2} / \binom{n}{2}
\]

(1.15)

Where \( n \) is the total number of objects, \( n_{ij} \) is the number of object partitions \( u_i \) and \( v_j \) have in common, \( n_i \) is the number of objects in partition \( u_i \) and \( n_j \) is the number of objects in partition \( v_j \). These can be found easily with the contingency table:

<table>
<thead>
<tr>
<th>( U \setminus V )</th>
<th>( v_1 )</th>
<th>\ldots</th>
<th>( v_C )</th>
<th>Sums</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_1 )</td>
<td>( n_{11} )</td>
<td>\ldots</td>
<td>( n_{1C} )</td>
<td>( n_1. )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>\ldots</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( u_R )</td>
<td>( n_{R1} )</td>
<td>\ldots</td>
<td>( n_{RC} )</td>
<td>( n_R. )</td>
</tr>
<tr>
<td>Sums</td>
<td>( n_1 )</td>
<td>\ldots</td>
<td>( n_C )</td>
<td>( n )</td>
</tr>
</tbody>
</table>
Chapter 2

Community detection in graphs

2.1 Overview

The field of community detection in graphs is so vast, that Fortunato [9] needed a hundred pages to write a full review. The method used in this Master thesis has been chosen because it is a promising recent technique, being constantly improved, and the people working on it were easily accessible to me.

Traditional methods for graph clustering includes graph partitioning, in which groups of nodes are separated in a way that the number of edges joining the communities is minimal. Spectral clustering is another class of methods, in which the eigenvectors of the Laplacian of the graph are exploited.

Other methods are based on the measure of modularity. For example, Blondel et al. [3] proposed in 2008 an agglomerative hierarchical algorithm to detect communities in graphs, based on the measure of modularity, which outperformed other algorithms in terms of results quality and computation time.

The method explained below belongs to the class of algorithms that defines a distance measure between the nodes of the graph. Distance-based clustering algorithms can then be applied.

2.2 Commute-time kernel

The similarity measure used in this master thesis to cluster graphs has been developed in Fouss et al. [10], Saerens et al. [30] and Fouss et al. [12]. Using the pseudoinverse of the Laplacian of the graph, it is possible to compute the average
Chapter 2 : Community detection in graphs

commute-time, a distance which correspond to the average number of steps a random walker needs to go from node $i$ to node $j$, and back. We will start here by defining the random walker, then introduce the Laplacian of a graph. Interested readers may want to look for the previously cited papers, which contains much more details than the following.

2.2.1 Random walker

We assume we are working on a weighted and undirected graph. Hence, the adjacency matrix $A$ is symmetric. The random walk is defined as (Gobel and Jagers [14]):

A random walk on graph $G$ starting at vertex $i$ is a random sequence $x_0, x_1, x_2, ...$ of vertices of $G$ such that:

$$P(x_0 = i) = 1,$$

$$P(x_{m+1} = k | x_m = j) = \frac{a_{ij}}{a_i}.$$

Where $a_i = \sum_{j=1}^{n} a_{ij}$ The random walker starts from node $x_0$, and randomly chooses the next node to visit, with a probability proportional to the weight of the edge between the two nodes.

This random walker can be seen as a Markov chain. Indeed, a Markov chain is caracterised by a set of states $S = \{s_1, s_2, ..., s_n\}$ and transition probabilities $p_{ij}$ between state $i$ and $j$ which depend only on the current state of the system (Grinstead and Snell [15, chap. 11]).

2.2.2 Average commute-time

From this random walker, we can derive what is called the average first-passage time $m(k|i)$, which is the average number of steps the random walker needs to reach the node $i$ when starting from node $k \neq i$.

(Fouss et al. [10]) gives the recurrence relations allowing to compute $m(k|i)$:

$$\begin{align*}
  m(k|k) &= 0 \\
  m(k|i) &= 1 + \sum_{j=1; j \neq k}^{n} p_{ij} m(k|j) \text{ for } i \neq k
\end{align*}$$

The average commute-time is:
\[ n(i, j) = m(j|i) + m(i|j) \]

As stated before, it is the average number of steps the random walker needs to go from node \( i \) to node \( j \), and back. We do not explain how these formulas can be found, simply because they will not be used. The pseudoinverse of the Laplacian allows to compute these quantities without the need of recurrence.

It should be noted that, unlike the shortest path distance (which is also used in graph clustering), the average commute-time distance has the property to decrease when the number of paths connecting the two nodes increases, or when the length of these decreases.

### 2.2.3 Commute-time kernel

The Laplacian of a graph is defined as (Chung [8]):

\[
L(u, v) = \begin{cases} 
  d_v & \text{if } u = v, \\
  -1 & \text{if } u \text{ and } v \text{ are adjacent,} \\
  0 & \text{otherwise} 
\end{cases}
\]

Where \( d_v \) is the degree of node \( v \). It is actually the difference between the matrix of the degrees of \( A \) (\( D_{ii} = d_i \) and \( D_{ij} = 0 \) when \( i \neq j \)) and the adjacency matrix:

\[ L = D - A \]

Fouss et al. [10] shows that the average commute-time can be computed with the Moore-Penrose pseudoinverse of \( L \), \( L^+ \) (\( L \) is not invertible) with the formula:

\[ n(i, j) = V_G(e_i - e_j)^T L^+(e_i - e_j) \]

Where \( V_G \) is the volume of the graph (\( \sum_{k=1}^n d_{kk} \)) and \( e_i \) is a vector of zeros, except at the \( i \)-th position which contains the value 1.

The elements of \( L^+ \) have the interesting property of being the inner products of the node vectors in an Euclidian space where they are separated by the commute-time distance (Fouss et al. [11]). Naturally, the commute-time kernel is defined as:

\[ K_{CT} = L^+ \]
2.2.4 Regularized commute-time kernel

Instead of computing the Moore-Penrose pseudoinverse of $L$, we can regularize the Laplacian by adding a small quantity $\frac{1-\alpha}{\alpha}D$ to $D - A$ to make it invertible. We obtain the regularized commute-time kernel (Fouss et al. [11]):

$$K_{RCT} = (D - \alpha A)^{-1}$$  \hspace{1cm} (2.1)

With $\alpha \in [0, 1]$.

An interpretation of $\alpha$ is that, this time, the random walker has a $(1 - \alpha)$ probability of stopping the walk at each step. Fouss et al. [11] also provides another interpretation of the elements of the kernel. They can be seen as the probability of reaching node $j$ when starting from node $i$ (computed with the Markov chain of the random walker), with $\alpha$ decreasing the importance of later visits.

This kernel will be used along with the kernel k-means for graph clustering.

2.3 Spring embedder

Because it is always reassuring to see what our algorithm is actually doing, some time was spared to implement a graph visualization method.

A simple algorithm was introduced by Eades in 1984. It replaces the edges of the graph with springs of unit length, and join non-adjacent nodes with springs of infinite length. The nodes are initially placed randomly in the 2-dimensional space. Then, they are released. The forces applied on each node are computed using a variant of the Hookes’ Law (Kobourov [19]).

The attractive force between node $i$ and $j$ is computed with the formula:

$$f_a = c_1 \cdot \log\left(\frac{d_{ij}}{c_2}\right)$$

Where $c_1$ and $c_2$ are constants and $d_{ij}$ is the distance between $i$ and $j$. A logarithmic strength is used because the linear law of Hookes produces strings too strong.

The repulsive force is:

$$f_r = \frac{c_3}{\sqrt{d_{ij}}}$$

Where $c_3$ is also a constant.
Chapter 3

Textmining

Textmining concerns the extraction of useful information from text data. In particular, build some representation of text documents that can be used to cluster them. We will present here the steps allowing to construct a word index and the vector space model which will be used to cluster Wikipedia articles.

3.1 Word index of documents

A word index consist of the list of words that a collection of documents contain, along with the number of times each word appears in each document (the term frequency).

Since we used the Lucene library to build the word index, the following paragraphs gives only a general idea of the vast problem of word indexing. They are inspired from the book of Manning et al. [23, chap. 2].

To build a useful index, some processing is necessary. First of all, the tokenization divides the text into tokens, using either fixed rules or a dictionary. This correspond to divide the text into words, but it is actually more complex. For example, the sequence [aren’t] could be tokenized as [aren’t], [arent], [are] [n’t] or [aren] [t]. The tokenization also removes punctuation and other special characters. The presence of multiple languages makes this step even more tricky, as a language identifier must be used since every language has its own particularities.

The next step is the removal of stop words. Indeed, words as to, as, not, or have a high frequency of appearance, on any text document, and thus does not bring useful information.
Subsequently, the token normalization transforms words that can be written in different ways to their standard form. For example, by removing capitalization (Door and door), removing hyphens or more specific cases as colour (British) and color (American). Of course this brings problems like normalizing a person’s name Black into the colour black.

Finally, stemming or lemmatization is applied. The idea is to keep only the root (or stem) of each word. The fact that the verb be is conjugated as am, are or is has no importance. We just want to know that the verb be appears in the document. The difference between stemming and lemmatization is that stemming removes the end of each word (using some rules) (difference, differentiation, different all become differ) while lemmatization uses dictionaries for more precision.

### 3.2 Vector space model

Once the index of words is built, we can compute the needed statistics. Recall that the word index contains the frequency of each term in each document, formally:

\[
\text{tf}_{ij} = \frac{n_{ij}}{\sum_k n_{kj}}
\]

Where \(n_{ij}\) is the number of times the term \(i\) appears in document \(j\), and \(\sum_k n_{kj}\) is the total number of words in document \(j\).

The inverse document frequency is:

\[
\text{idf}_i = \log \frac{|D|}{df_i}
\]

Where \(|D|\) is the number of documents, and \(df_i\) is the document frequency, the number of documents where the term \(i\) appears.

We can combine these two values and compute the tf-idf weight:

\[
\text{tf-idf}_{ij} = \text{tf}_{ij} \cdot \text{idf}_i
\]  \hspace{1cm} (3.1)

The \(\text{tf-idf}_{ij}\) is high when the term \(i\) appears with a high term frequency in document \(j\) and has a low document frequency among the collection. Thus, for each document, the \(\text{tf-idf}\) vector represents which words makes it different with respect to the others.
3.3 Cosine similarity

The most common method used to define a similarity measure between documents is by computing the cosine between the vectors of tf-idf values.

Since the cosine is the scalar product divided by the norm of both vectors, we can build a kernel matrix with the scalar products, and then normalize the kernel with formula 1.13.

The kernel is then used with the spherical kernel k-means to cluster the documents.

3.4 Lucene

To build the word index of Wikipedia articles, the Lucene 3.0.3 Java library was used. This open source library provides all the necessary functions to build indexes and process search queries.

The Snowball Analyzer was used to create the index. This object successively tokenizes, normalizes, filters upper-case characters, removes english stop words and applies stemming to the documents. The tf, idf and tf-idf are then computed with the preceding formulas.

\footnote{Many thanks to Joachim de Beule for providing his implementation of an index builder, which was slightly adapted for the present work}
Chapter 4

Wikipedia

As many biological or social networks, the graph of Wikipedia articles has been found to be a small world graph. Small world graphs are characterized by a high clustering coefficient (there are clumps of nodes highly linked) and a low characteristic path length (all the nodes are close to each other) (Watts and Strogatz [33]).

Furthermore, Capocci et al. [7] measured the degree distribution of the directed graph of articles, and showed that it is scale free (the degree distribution follows a power law). The scale free structure of graphs, while seemingly complex, can actually be explained by the simple phenomenon of preferential attachment, introduced by Barabasi and Albert [1]. The graph is built one node at a time, and the new node attaches preferentially to nodes that are already highly connected. Capocci et al. [7] suggested that the scale free structure of Wikipedia articles is either a natural organization of knowledge, or a result of the contributors behavior, who prefer to concentrate on writing articles than efficiently organize the data. In other words, people would link articles to main nodes of the graph because it is easier than looking through all the available articles.

Other algorithms producing scale free networks exist, though. Caldarelli et al. [6], for example, define a method where the graph is static (as opposed to the preferential attachment, where the graph grows) and the probability of two nodes being linked depends on a fitness value, intrinsic to each node. This applies well to Wikipedia, since the degree of the nodes are not known by the contributors. Instead, some articles are more important, central or main.

Since we will focus on small parts of the Wikipedia graph, a method has been set up to extract the datasets. The following section presents it.
Chapter 4 : Wikipedia

4.1 Building datasets

Wikipedia dumps are available online\(^1\), and regularly updated. The files used to build the datasets date from February 2011. The English dumps were used.

To parse most of the files, the Parse::MediaWikiDump 1.0.6 Perl library was used. This allows to easily extract the data from the xml and sql files.

Before performing the extraction of datasets, we need the graph of categories and articles.

The content, id’s, titles and categories of each article are contained in the file enwiki-latest-pages-articles.xml.

The links between articles can be build from the file enwiki-latest-pagelinks.sql. This file maps the article id’s to the title of the linked article. This means that one needs to know the corresponding id for each title. Also, sometimes, article titles simply do not exist, because the page was deleted or renamed. Disambiguations were not considered (the link was dropped). The links to redirect pages were modified to link to the correct page right away.

Building the graph of categories is more complex. The file enwiki-latest-page.sql contains the id, namespace and title of all the pages. A page can be an article, a category, a portal, etc. The namespace allows to filter everything but the categories.

Once the id of each category is known, the links between categories can be extracted from enwiki-latest-categorylinks.sql. This file contains each category and all the categories to which it belongs (supercategories). The file maps the categories id’s to the titles of the supercategories. Again, some categories may not exist anymore. And also, note that MediaWikiDump is not able to parse these two last files.

The method established to extract categories is explained in figure 4.1.

The graph of categories is directed. Each category has subcategories and supercategories. There are cycles, and sometimes a category will have main categories as childs (as Mathematics, Technology or Philosophy). Thus, by listing all the subcategories, we end up with the whole database. To narrow the search, a first filter was implemented to keep categories having maximum 500 subcategories.

The graph of articles is originally directed, but was transformed to be undirected and unweighted\(^2\). Two articles are linked when at least one has an hyperlink to the other.

\(^1\)http://dumps.wikimedia.org/enwiki/latest/

\(^2\)To make the graph unweighted was a mistake, since the regularized commute-time can handle weighted graphs. The weight of the link could have been the number of links between the articles.
towards the other. Another filter was implemented, to keep only connected components of size ranging from 500 to 20,000 articles (for a total of nearly 24,000 datasets).

Among the extracted categories, a few with interesting sizes and names were chosen and are analysed in section 5.2.
On the graph of categories, a category is selected. All the subcategories (until reaching the leafs) are listed.

On the articles graph, all the articles belonging to the listed categories are selected. Only the links among the selected articles are kept.

The largest connected component is selected. The extracted graph for the category c1.
Chapter 5

Experiments

The experimental part of this master thesis is divided in two sections. First, the performance of different parameterizations of the commute-time is assessed. Then, Wikipedia datasets are clustered and the results discussed.

5.1 Effects of center and normalize

To assess the influence on the performance of centering and normalize the regularized commute-time kernel, we will run the algorithm on four well known graphs. This will also allow to determine which values of the parameter $\alpha$ gives better results. Also, a comparison with the sigmoid commute-time kernel introduced in Yen et al. [34] will be possible.

We will first present the four graphs, then describe the parameters of the experiment and finally analyse the results.

An important recall is that all the graphs are undirected.

5.1.1 Zachary’s karate club

The Zachary’s karate club was observed by Zachary [35] for three years, from 1970 to 1972. It consists of 34 members of the club and their social relationships. Two members are linked in the graph if they were observed interacting outside of the club (friendship). The interesting aspect of the graph is that the club was subject to political crisis. From the viewpoint of each individual, there was no sentiment of division. However, at each confrontation between the karate instructor (a spiritual mentor for his supporters) and the administrator of the club (seeing the
instructor as an employee trying to obtain a higher salary), Zachary could observe two clearly separated factions. The strain of the conflict increased until the karate instructor was fired, and created with his supporters a new organization.

The goal here is then to find the two factions by clustering the graph. The figure 5.1 shows the graph.

5.1.2 Dolphin Network

The dolphin network is a graph created by Lusseau et al. [21]. For seven years, from 1994 to 2001, Lusseau and his associates observed dolphins schools in Doubtful Sound, New Zealand, identifying each dolphin with photographs of their dorsal fins. Associations in the graph are defined through a statistical study. Dolphins are linked when they were observed together more often than expected by chance. An interesting phenomenon occurred when an important member disappeared during the study. The two communities highlighted in 5.2 reduced their interactions until the dolphin returned (Lusseau and Newman [20]). These are then the communitites we will look for during clustering.
5.1.3 IMDB

The IMDB dataset originates from Neville and Jensen [27]. Using data from the online movie database, Neville built a predictor of whether a new movie would reach 2 million dollars of receipts during the opening week-end. The dataset, incorporating attributes of the movies like the actors, directors, producers, studios, was reprocessed by Macskassy and Provost [22] who made a graph with 1169 movies released in the United States between 1996 and 2001. The movies are linked if they share a production company. The weight of the link is the number of companies they have in common.

Two communities can be defined: the movies whose receipts exceeded 2 million dollars during the opening week-end, and the others. We worked here on the largest fully connected subgraph, which contains 1126 movies.

5.1.4 CORA

The Cora portal was designed by Mccallum et al. [25] as an online search system for computer science papers. Macskassy and Provost [22] used the available data (the citation graph and topic labels) to build a graph of 3583 nodes. The papers are linked when one cites the other. Each paper belongs to one topic, which can be Case-based, Genetic Algorithms, Neural Networks, Probabilistic Methods,
Chapter 5: Experiments

Reinforcement Learning, Rule Learning, Theory. There are seven topics, thus seven clusters.

5.1.5 Visualization of the kernels

Before running the clustering, we can compute the regularized commute-time kernel for the Zachary’s karate club and visualize it, using the function `imagesc` of Octave (or Matlab). This function will build a color scale from the lowest value of the matrix (blue) to the highest (red), and color each element with a color corresponding to his value. Light blue and yellow are intermediate values. Figure 5.3 shows the obtained images, for values of $\alpha$ ranging from 0.1 to 0.9.

![Figure 5.3: Kernels of Zachary’s karate club.](image)

The rows and columns of the kernel were sorted so that the nodes of each cluster are together (top-left and bottom-right). Recall that the values of the kernel are the scalar products between the vectors. The normalized kernels contains the values of the cosine, which is used as the measure of similarity in spherical k-means. Light blue, yellow and red means high similarity between the nodes (indeed, the diagonals are always red). However, for the standard kernel, the values are part of the formula $\|u - v\|^2 = u.u - 2u.v + v.v$. The image does not represent the similarity between the nodes. We can nevertheless observe the evolution of the kernel with the parameter $\alpha$. 

What we can conclude from these images is that the higher the value of $\alpha$, the clearer the difference between the two clusters. Higher values of $\alpha$ should give better clustering results. Besides, normalized and centered-normalized kernels show good partition of the data. The results of clustering are presented below.

5.1.6 Experiment settings

![Graphs showing clustering with the regularized commute-time kernel.](image)

Figure 5.4: Clustering with the regularized commute-time kernel. Standard (black), normalized (blue) and centered-normalized (green). The mean of the adjusted rand index among 30 runs (of 50 iterations) is plotted versus $\alpha$ values ranging from 0.05 to 0.95.

The kernel k-means algorithm was runned with the regularized commute-time kernel (standard, normalized, and centered-normalized) on these four graphs, with
α equal to values between 0.05 and 0.95. The algorithm was iterated 50 times, and the best result, corresponding to the lowest (highest) objective function value was kept. The adjusted rand index was used to compare the solution to the labels. Moreover, the experience was runned 30 times. The mean of the adjusted rand index among these 30 runs is showed in figure (5.4).

5.1.7 Discussion

The first observation is that, as expected, the quality of the solution always improves by increasing α. The value 0.95 gives the best result for the four graphs. For the IMDB dataset, increasing even more the α value improves the solution, but not significantly. For α = 0.995, the adjusted Rand index reach 0.290, while it is 0.276 at α = 0.95.

Moreover, center and normalize the kernel is the best strategy for Karate Club, IMDB and CORA. For the Dolphins Network, at α = 0.95, not centering the kernel allows to correctly classify one last node (positioned at the edge of the two clusters). However, the results of the normalized and the centered-normalized kernel are always close, the latter giving slightly better results on average.

The quality of IMDB and CORA is poor. For the IMDB dataset, this can be explained by the structure of the graph itself. Indeed, the graph only contains information about the production companies of the movies. This is clearly not enough to determine if the movie will be a success or not.

We can make an interesting comparison with the variant of the commute-time kernel introduced in Yen et al. [34]. The sigmoid commute-time applies a sigmoid function to the values of the kernel. This has proved useful in reducing the contribution of outliers, and to improve results overall. The table 5.1 compares the best results of Yen et al. [34] with the ones found here. The settings of the experiment where the same (30 runs of 50 iterations). (\(K_{RCT}\)) stands for centered and normalized regularized commute-time kernel. The sigmoid commute-time performs better on CORA, but has the same performance on IMDB.

<table>
<thead>
<tr>
<th></th>
<th>IMDB ((\alpha = 0.995))</th>
<th>CORA ((\alpha = 0.95))</th>
</tr>
</thead>
<tbody>
<tr>
<td>((K_{RCT})_{cn})</td>
<td>0.29</td>
<td>0.30</td>
</tr>
<tr>
<td>(K_{CT}^S)</td>
<td>0.29</td>
<td>0.46</td>
</tr>
</tbody>
</table>

Table 5.1: Comparison of the centered-normalized regularized commute-time and the sigmoid commute-time.
All the following experiments will be using the centered-normalized kernel, with $\alpha = 0.95$.

5.2 Clustering Wikipedia

Now that the clustering algorithm is fine tuned, we can apply it to the extracted Wikipedia datasets from section 4.1.

For the graph, the kernel used is the centered-normalized regularized commute-time kernel ($\alpha = 0.95$). For the word index, the kernel containing the cosine similarity between the tf-idf vectors is used.

The spherical kernel k-means is restarted 50 times with a random initialization, and the solution with the highest objective value is kept.

The Wikipedia articles belong to multiple categories at the same time. There are no clear classes, so there is no good clustering solution that we can use as comparison. We will thus assess the quality of the results by looking at the article titles of each cluster.

Since no method to detect the optimal number of clusters has been implemented, these will be estimated based on the type and characteristics of the category (sometimes, multiple trials were necessary, the following sections only shows the best obtained results).

5.2.1 Cycleways

Figure 5.5: Cycleways graph. For visibility purposes, the edges are not printed.
Cycleways contains 3 main subcategories, Cycleways in Europe, Cycleways by country and Rail trails. There are 571 articles in total.

<table>
<thead>
<tr>
<th>Cluster 1 (109 articles)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>List of cycleways</td>
<td></td>
</tr>
<tr>
<td>Cycleways in England</td>
<td></td>
</tr>
<tr>
<td>List of rail trails in Minnesota</td>
<td></td>
</tr>
<tr>
<td>Airline State Park</td>
<td></td>
</tr>
<tr>
<td>List of rail trails in West Virginia</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cluster 2 (104 articles)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Great Allegheny Passage</td>
<td></td>
</tr>
<tr>
<td>Ohio River Trail</td>
<td></td>
</tr>
<tr>
<td>Panhandle Trail</td>
<td></td>
</tr>
<tr>
<td>York County Heritage Rail Trail</td>
<td></td>
</tr>
<tr>
<td>List of rail trails in Pennsylvania</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cluster 3 (100 articles)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Van Buren Trail State Park</td>
<td></td>
</tr>
<tr>
<td>White Pine Trail State Park</td>
<td></td>
</tr>
<tr>
<td>Kent Trails</td>
<td></td>
</tr>
<tr>
<td>Leelanau Trail</td>
<td></td>
</tr>
<tr>
<td>Pere Marquette Rail-Trail</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cluster 4 (143 articles)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rosstown Railway Heritage Trail</td>
<td></td>
</tr>
<tr>
<td>Outer Circle Trail</td>
<td></td>
</tr>
<tr>
<td>Ringwood–Belgrave Rail Trail</td>
<td></td>
</tr>
<tr>
<td>Sandridge Trail</td>
<td></td>
</tr>
<tr>
<td>Capital City Trail</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cluster 5 (115 articles)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Lilydale to Warburton Rail Trail</td>
<td></td>
</tr>
<tr>
<td>Bass Coast Rail Trail</td>
<td></td>
</tr>
<tr>
<td>Great Southern Rail Trail</td>
<td></td>
</tr>
<tr>
<td>Say G’day Rail Trail</td>
<td></td>
</tr>
<tr>
<td>Old Beechy Rail Trail</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: 5 most important articles of each cluster, Cycleways. In green, the category Rail trails in Pennsylvania. In blue, Rail trails in Michigan. In yellow, Bike paths in Melbourne. In magenta, Rail trails in Victoria.

The figure 5.5 shows the result of the clustering, with 5 clusters. The coordinates of each node are computed with the spring embedder, but note that since the graph is highly dense, the edges are not printed.

The visualization of the clustering shows that the articles are clearly separated (even the spring embedder found the groups). Table 5.2 shows the 5 most important articles of each cluster. The important criterion is the degree of the node.

Without knowing about rail tracks around the world, one would say that the articles are randomly classified. However, by looking at the subcategories of Cycleways, we find a pattern. Cluster 2 contains, among others, all the articles from
Chapter 5: Experiments

Rail trails in Pennsylvania. Cluster 3, from Rail trails in Michigan. Cluster 4, from Bike paths in Melbourne. Cluster 5, almost all the articles from Rail trails in Victoria. Cluster 1, on the other hand, holds articles from multiple subcategories. Note that the colors of the table correspond to the figure 5.5.

This is an interesting result. The categories of each article are chosen by the people writing them. The same applies to the links. We find here that articles inside the same category are strongly linked together. A possible explanation is that, when someone writes a new article, he will look at other articles among the closest categories to place links and make the article known.

We further continue the analysis with another category.

5.2.2 Chess history

Chess history contains 917 articles. We can find two main kinds of articles: the chess players, and the chess competitions. Figure 5.6 shows the obtained clustering with 2 clusters.

Figure 5.6: Chess history graph. Result of graph clustering. (The edges are not printed).

This results contrast with the previous. Here, we don’t immediately see groups of nodes together. The table 5.3 shows that the first cluster has a majority of chess players, but the second has chess players and Championships or Olympiads.

Indeed, by looking at the articles, all the chess players’ pages contain links to the FIDE (Fédération Internationale des Échecs) and the competitions. On every Championship page, all the participants are listed, with a link to their page.
Table 5.3: 10 most important articles of each cluster, Chess history. Graph clustering.

<table>
<thead>
<tr>
<th>Cluster 1 (480 articles)</th>
<th>Cluster 2 (437 articles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chess Olympiad</td>
<td>FIDE titles</td>
</tr>
<tr>
<td>World Chess Championship</td>
<td>Vladimir Kramnik</td>
</tr>
<tr>
<td>Alexander Alekhine</td>
<td>Viswanathan Anand</td>
</tr>
<tr>
<td>Bobby Fischer</td>
<td>Mikhail Botvinnik</td>
</tr>
<tr>
<td>Garry Kasparov</td>
<td>Boris Spassky</td>
</tr>
<tr>
<td>Anatoly Karpov</td>
<td>FIDE World Chess Championship 2002</td>
</tr>
<tr>
<td>Vasily Smyslov</td>
<td>FIDE World Chess Championship 1999</td>
</tr>
<tr>
<td>Max Euwe</td>
<td>Paul Keres</td>
</tr>
<tr>
<td>José Raúl Capablanca</td>
<td>World Chess Championship 1972</td>
</tr>
<tr>
<td>Mikhail Tal</td>
<td>6th Chess Olympiad</td>
</tr>
</tbody>
</table>

Evidently, the random walker has difficulties partitioning such a highly connected graph.

This is when the word index becomes useful. Figure 5.7 shows the results obtained by clustering the tf-idf vectors of the documents, using the cosine similarity. Table 5.4 shows the most important articles for each cluster.

Figure 5.7: Chess history graph. Bag of words clustering. (The edges are not printed).

The chess players and the Championships/Olympiads are now much better separated. The interesting observation is on figure 5.7, where we can see that the two clusters are overlapped. This explains why the random walker is not able to partition them correctly.

The word index can thus, sometimes, perform better than the graph. The whole problem is to know when. As a side note, Mantrach et al. [24] successfully
used the graph data along with the word index data, and obtained an increase in performance for classification. This confirms that using both informations can be beneficial.

5.2.3 Fashion accessories

Fashion accessories contains 3080 articles in total. Main subcategories concerns the Headgear, Watches and Jewellery. Other smaller subcategories contains various kinds of accessories, from the Belts and Eyewears to Ceremonial weapons.

A clustering with 8 clusters was launched. The figure 5.8 shows the results for the links and index clusterings. By looking at the most important articles for each cluster (table available in Appendice A), some clusters could be identified.

We start by observing the links clustering. The partition seems clean and logical. Two clusters (the fuchsia and pink) could not be identified on the picture because the nodes are too small. The first contains diamond articles while the later is more fuzzy, with articles from metalsmithing, jewellery and other accessories.

The blue cluster is a strange one. What does the resurrection of Jesus have to do with fashion accessories? The answer lies within Prayer beads, which leads to Rosary, the Mysteries of the Rosary and finally the Resurrection of Jesus. The random walker successfully separated these articles from the rest.

The clustering of the word index gives less precise results. The watches and the crowns have been grouped together. The beads, metal accessories and the resurrection of Jesus are also together. Finally, the hats, handbags, sunglasses have been splitted into two clusters with mixed articles.

<table>
<thead>
<tr>
<th>Cluster 1 (482 articles)</th>
<th>Cluster 2 (435 articles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grandmaster (chess)</td>
<td>FIDE World Chess Championship 2002</td>
</tr>
<tr>
<td>World Chess Championship</td>
<td>FIDE World Chess Championship 1998</td>
</tr>
<tr>
<td>FIDE titles</td>
<td>FIDE World Chess Championship 2000</td>
</tr>
<tr>
<td>Alexander Alekhine</td>
<td>FIDE World Chess Championship 1999</td>
</tr>
<tr>
<td>Bobby Fischer</td>
<td>4th Chess Olympiad</td>
</tr>
<tr>
<td>Garry Kasparov</td>
<td>2nd Chess Olympiad</td>
</tr>
<tr>
<td>Anatoly Karpov</td>
<td>36th Chess Olympiad</td>
</tr>
<tr>
<td>Vasily Smyslov</td>
<td>37th Chess Olympiad</td>
</tr>
<tr>
<td>Max Euwe</td>
<td>6th Chess Olympiad</td>
</tr>
<tr>
<td>José Raúl Capablanca</td>
<td>World Chess Championship 2007</td>
</tr>
</tbody>
</table>

Table 5.4: 10 most important articles of each cluster, Chess history. Bag of words clustering.
But we also get more information about the structure of the data. The Sierra Leone civil war (which is linked to fashion accessories through the blood diamonds category) was detected, while the random walker grouped it with the gemstones. The index also separated the easter eggs from the religion articles and could divide the luxury accessories into gemstones and metal-based.

The results are then mixed, compared to the two previous datasets. Which partition is the best must be discussed. This shows all the difficulty of clustering. We can nevertheless observe that the clusterings found here have similarities with the categories of Wikipedia. The articles from closely related categories contain similar words, but are also naturally strongly linked together.
Figure 5.8: Results of the clustering of Fashion Accessories.
Conclusion

The first result of this Master thesis is the considerable increase of clustering performance when the regularized commute-time kernel is centered and normalized. Results come even close to the sigmoid commute-time kernel.

Secondly, the results show that a rational partition of the articles of Wikipedia can be found by clustering the graph of links. It is however not always possible, as some categories are so strongly linked that only the word index allows to correctly partition them. One needs however to keep in mind that many links were removed during the extraction process, and thus these results apply only to subsets of the graph.

Finally, intuition says that larger datasets will become increasingly difficult to partition using the graph data, as the small world properties of the network will begin to be felt.

After a year learning about graphs, I feel like I have only scratched the surface of the field. There are thousands of articles about this subject, each of them bringing new insights, new ideas and new discoveries. This Master thesis was a great experience, an insight into the world of scientific research that will for sure be remembered.
Bibliography


## Appendix A

### Fashion accessories

<table>
<thead>
<tr>
<th>Cluster 1 (470 articles)</th>
<th>Cluster 2 (258 articles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hat</td>
<td>Beadwork</td>
</tr>
<tr>
<td>Helmet</td>
<td>Glass beadmaking</td>
</tr>
<tr>
<td>Headgear</td>
<td>Millefiori</td>
</tr>
<tr>
<td>Cap</td>
<td>Bead crochet</td>
</tr>
<tr>
<td>Sunglasses</td>
<td>Sequin</td>
</tr>
<tr>
<td>Veil</td>
<td>Murano beads</td>
</tr>
<tr>
<td>Fashion accessory</td>
<td>Chevron bead</td>
</tr>
<tr>
<td>Button</td>
<td>Bead embroidery</td>
</tr>
<tr>
<td>Handbag</td>
<td>Faturan</td>
</tr>
<tr>
<td>Hatmaking</td>
<td>Trade beads</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cluster 3 (390 articles)</th>
<th>Cluster 4 (289 articles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easter egg</td>
<td>Mask</td>
</tr>
<tr>
<td>Resurrection of Jesus</td>
<td>Vacheron Constantin</td>
</tr>
<tr>
<td>Crucifixion of Jesus</td>
<td>Jaeger-LeCoultre</td>
</tr>
<tr>
<td>Last Supper</td>
<td>International Watch Company</td>
</tr>
<tr>
<td>Pentecost</td>
<td>Baume et Mercier</td>
</tr>
<tr>
<td>Palm Sunday</td>
<td>Montblanc (company)</td>
</tr>
<tr>
<td>Lent</td>
<td>Piaget SA</td>
</tr>
<tr>
<td>Great Lent</td>
<td>A. Lange &amp; Söhne</td>
</tr>
<tr>
<td>Dormition of the Theotokos</td>
<td>Richemont</td>
</tr>
<tr>
<td>Holy Saturday</td>
<td>Panerai</td>
</tr>
</tbody>
</table>

Table A.1: Links clustering, part 1.
### Table A.2: Links clustering, part 2.

<table>
<thead>
<tr>
<th>Cluster 5 (286 articles)</th>
<th>Cluster 6 (433 articles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goldsmith</td>
<td>Watchmaker</td>
</tr>
<tr>
<td>Silversmith</td>
<td>List of watch manufacturers</td>
</tr>
<tr>
<td>Engraved gem</td>
<td>List of watch manufacturers</td>
</tr>
<tr>
<td>Tiffany &amp; Co.</td>
<td>Glasses</td>
</tr>
<tr>
<td>Solder</td>
<td>ETA SA</td>
</tr>
<tr>
<td>Fibula (brooch)</td>
<td>Chronograph</td>
</tr>
<tr>
<td>Art jewelry</td>
<td>Omega SA</td>
</tr>
<tr>
<td>Brazing</td>
<td>Rolex</td>
</tr>
<tr>
<td>Artisan</td>
<td>Seiko</td>
</tr>
<tr>
<td>Jewellery Quarter</td>
<td>COSC</td>
</tr>
<tr>
<td><strong>Cluster 7 (282 articles)</strong></td>
<td><strong>Cluster 8 (672 articles)</strong></td>
</tr>
<tr>
<td>List of diamond mines</td>
<td>Gemstone</td>
</tr>
<tr>
<td>Diamonds as an investment</td>
<td>Crown (headgear)</td>
</tr>
<tr>
<td>Harry Winston</td>
<td>Quartz</td>
</tr>
<tr>
<td>Kimberlite</td>
<td>Carat (mass)</td>
</tr>
<tr>
<td>Brown diamonds</td>
<td>Sapphire</td>
</tr>
<tr>
<td>List of largest rough diamonds</td>
<td>Necklace</td>
</tr>
<tr>
<td>Randlord</td>
<td>Pearl</td>
</tr>
<tr>
<td>Cecil Rhodes</td>
<td>Vitreous enamel</td>
</tr>
<tr>
<td>American Gem Society</td>
<td>Ruby</td>
</tr>
<tr>
<td>Premier Mine</td>
<td>Emerald</td>
</tr>
</tbody>
</table>

### Table A.3: Word index clustering, part 1.

<table>
<thead>
<tr>
<th>Cluster 1 (933 articles)</th>
<th>Cluster 2 (594 articles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goldsmith</td>
<td>Gemstone</td>
</tr>
<tr>
<td>Necklace</td>
<td>Quartz</td>
</tr>
<tr>
<td>Vitreous enamel</td>
<td>Carat (mass)</td>
</tr>
<tr>
<td>Earring</td>
<td>Sapphire</td>
</tr>
<tr>
<td>Easter</td>
<td>Ruby</td>
</tr>
<tr>
<td>Onyx</td>
<td>Emerald</td>
</tr>
<tr>
<td>Belly chain</td>
<td>Amethyst</td>
</tr>
<tr>
<td>Jade</td>
<td>Opal</td>
</tr>
<tr>
<td>Engraving</td>
<td>Agate</td>
</tr>
<tr>
<td>Jasper</td>
<td>Gemology</td>
</tr>
<tr>
<td><strong>Cluster 3 (230 articles)</strong></td>
<td><strong>Cluster 4 (129 articles)</strong></td>
</tr>
<tr>
<td>Papal Tiara</td>
<td>Glove</td>
</tr>
<tr>
<td>Tiara</td>
<td>Sierra Leone Civil War</td>
</tr>
<tr>
<td>Sunglasses</td>
<td>Special Court for Sierra Leone</td>
</tr>
<tr>
<td>Glasses</td>
<td>Revolutionary United Front</td>
</tr>
<tr>
<td>Palatine Tiara</td>
<td>United Nations Mission in Sierra Leone</td>
</tr>
<tr>
<td>Mask</td>
<td>Ahmad Tejan Kabbah</td>
</tr>
<tr>
<td>The Personal Jewel Collection of Elizabeth II</td>
<td>Gauntlet (glove)</td>
</tr>
<tr>
<td>Decoration of the Papal Tiara</td>
<td>United Nations Security Council Resolution 1289</td>
</tr>
<tr>
<td>Contact lens</td>
<td>Armed Forces Revolutionary Council</td>
</tr>
<tr>
<td>Mitre</td>
<td>United Nations Security Council Resolution 1306</td>
</tr>
<tr>
<td>Cluster 5 (345 articles)</td>
<td>Cluster 6 (511 articles)</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Hat</td>
<td>Crown (headgear)</td>
</tr>
<tr>
<td>Ring (jewellery)</td>
<td>Watchmaker</td>
</tr>
<tr>
<td>Headgear</td>
<td>List of watch manufacturers</td>
</tr>
<tr>
<td>List of jewellery types</td>
<td>Crown jewels</td>
</tr>
<tr>
<td>Anklet</td>
<td>Crown Jewels of the United Kingdom</td>
</tr>
<tr>
<td>Chatelaine (chain)</td>
<td>Regalia</td>
</tr>
<tr>
<td>Tie clip</td>
<td>Pocket watch</td>
</tr>
<tr>
<td>Archducal Hat of Tyrol</td>
<td>Austrian Crown Jewels</td>
</tr>
<tr>
<td>Nipple shield (jewelry)</td>
<td>Regalia of the Russian tsars</td>
</tr>
<tr>
<td>Captive bead ring</td>
<td>Imperial State Crown</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cluster 7 (139 articles)</th>
<th>Cluster 8 (199 articles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collar pin</td>
<td>Belt buckle</td>
</tr>
<tr>
<td>Easter Bunny</td>
<td>Belt (clothing)</td>
</tr>
<tr>
<td>Here Comes Peter Cottontail</td>
<td>Rosary</td>
</tr>
<tr>
<td>Fabergé egg</td>
<td>Taqiyah (cap)</td>
</tr>
<tr>
<td>Egg hunt</td>
<td>Fez (hat)</td>
</tr>
<tr>
<td>Egg rolling</td>
<td>Šajkača</td>
</tr>
<tr>
<td>Traditional Easter games and customs</td>
<td>Balacava (clothing)</td>
</tr>
<tr>
<td>Egg decorating in Slavic culture</td>
<td>kkippah</td>
</tr>
<tr>
<td>Egg tapping</td>
<td>Suspenders</td>
</tr>
<tr>
<td>Pace Egg play</td>
<td>Gaiters</td>
</tr>
</tbody>
</table>

Table A.4: Word index clustering, part 2.