Practical challenges that arise when clustering the web using spectral methods

Argimiro Arratia[⋆1] and Carlos Marijuán[⋆⋆2]

[1] Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Spain, argimiro@lsi.upc.es
[2] Matemática Aplicada, Universidad de Valladolid, Spain, marijuan@mat.uva.es

Abstract. This is a report on an implementation of a spectral clustering algorithm for classifying very large internet sites, with special emphasis on the practical problems encountered in developing such a data mining system. Remarkably some of these technical difficulties are due to fundamental issues pertaining to the mathematics involved, and are not treated properly in the literature. Others are inherent to the functions and numerical methods proper to the high level technical computing programming environment that we use. We will point out what these practical challenges are and how to solve them.

Key words: spectral, clustering, internet

1 Introduction

Spectral clustering is a technique for partitioning data based on the spectrum of a similarity matrix: a matrix which registers some pairwise similarity between the data points, and to which one computes its *eigenvalues* and *eigenvectors* and uses these for clustering, or grouping, data as points in a smaller dimension. This data mining method seem to be gaining popularity, over other more traditional clustering algorithms, for its robustness and adequacy for effectively classifying large and sparse data sets.

The interest of the authors in understanding the topology of the Web [1,2], and the linear algebra behind spectral graph theory, led them to conduct a student's math and computation project for retrieving and clustering large parts of internet sites, with the main objective of understanding the methods rather than drawing conclusions from the particular classification.

Our main guide for learning about spectral clustering is the fine survey by Luxburg [11], and the fundamental papers of Shi and Malik [10], and Ng,

⋆ Research partially supported by Spanish Government MICINN under Projects: SESAAME (TIN2008-06582-C03-02) and SINGACOM (MTM2007-64007)
⋆⋆ Research partially supported by Spanish Government MICINN Project SINGACOM (MTM2007-64007)

Jordan and Weiss [9]. When we set about implementing the algorithms as explained in these references we ran into unforeseen troubles related to the convergence of the numerical methods used to compute eigenvalues for very large, and possibly sparse, matrices. One of these difficulties arises from the fact that the spectral problem, as stated in the consulted references on spectral clustering, is badly *conditioned*. We will explain what this means, and how to overcome this issue, which is surprisingly overlooked in the literature, although it is inherent to the computational methods involved in spectral clustering. Other difficulties are related with the limitations of the computational system that we used for implementing our algorithmic solution; in our case it is *MatLab* wired-up with C++. We will point out what these are and how to solve them. Also, faced with the challenge of exploring large parts of the internet in an efficient manner, we needed to get hold of a robust and powerful (and preferably, freely available) web crawler. Our selected system was WIRE [12], a professional open-source internet crawler designed for exploring and making statistics of usage of big communities' web sites (e.g. WIRE has been used for analyzing the full internet domain of Chile and Spain). As magnificent as WIRE can be, it needed some adaptation for our particular purposes, as for example, the addition of the extra functionality of generating the adjacency matrix of the explored domain in the format adequate for our computations. We will indicate the necessary changes and outline our practical steps for implementing a spectral clustering method.

We begin this article with a quick overview of spectral clustering. The reader should consult the survey [11] for more details. We then explain our small contributions and fixes for the benefit of those wishing to put into practice a spectral clustering algorithm successfully. We conclude with some comments on the results obtained from our experiments.

2 Spectral clustering in a nutshell

2.1 Clustering with respect to a similarity

We have a set of data points $V = \{v_1, \ldots, v_n\}$ and a similarity relation $s_{ij} = s(v_i, v_j) \geq 0$ on each pair v_i, $v_j \in V$. The goal of clustering V, with respect to the similarities $(s_{ij})_{1 \leq i,j \leq n}$, is to partition V into $k > 0$ subsets V_1, $\ldots$, V_k, such that any two points v_i, v_j in the same subset V_h have similarity $s_{ij} > C$, for some constant $C \geq 0$ (and we shall say that v_i, v_j are *closely* similar), and points in different subsets, $v_i \in V_h$ and $v_j \in V_{h'}$, $h \neq h'$, have similarity $s_{ij} \leq C$ (v_i, v_j are dissimilar).

The choice of similarity relation depends on the nature of the data and the statistical properties we would like to deduce from it. This is a large theme of research that we would prefer not to get into at present. For our experiments we chose as similarity measure the *Jaccard coefficient* between two sets, which is defined for two sets V_i and V_j as the quantity

$$jac(V_i, V_j) = \frac{|V_i \cap V_j|}{|V_i \cup V_j|} \tag{1}$$

2.2 Laplacian matrices

Now, given data set $V = \{v_1, \ldots, v_n\}$ and a similarity relation $S = (s_{ij})_{1 \leq i,j \leq n}$ on V, define for each $i = 1, \ldots, n$, the *volume* of v_i as the quantity $D_i = \sum_{j=1}^{n} s_{ij}$, and let $D = diag(D_1, \ldots, D_n)$ be a diagonal matrix of volumes.

There are several variances of Laplacian matrices, and they constitute in themselves a whole field of study (see e.g. [4]). In the context of spectral clustering there are three forms of interest [11]. For S and D as above, we have:

unnormalized Laplacian: $L = D - S$
normalized symmetric Laplacian: $L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{1/2} S D^{-1/2}$
normalized random walk Laplacian: $L_{rw} = D^{-1} L = I - D^{-1} S$

(Indeed, L_{sym} is a symmetric matrix, and L_{rw} is related to a random walk.)

Proposition 1 (Properties of L). *The Laplacian matrix L satisfies the following properties:*

1. *For every vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, we have*
   $x^t L x = \frac{1}{2} \sum_{1 \leq i,j \leq n} s_{ij}(x_i - x_j)^2$. *($x^t$ denotes the transpose of x.)*
2. *L is symmetric and positive semi-definite.*
3. *The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector $\mathbf{1}$.*
4. *L has n non negative, real-valued, eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.* □

The Laplacian matrices L_{sym} and L_{rw} satisfy similar properties. See [11] for details, and [8].

We close this section recalling some standard facts about matrices and their eigenvalues that we will need later on.

For any $n \times n$ real valued matrix, A, the *spectrum* of A, $\sigma(A)$, is the list of its eigenvalues; that is, $\sigma(A) = \{\lambda_1, \ldots, \lambda_n\}$. The *spectral radio* of A is $\rho(A) := \max\{|\lambda_i| : \lambda_i \in \sigma(A)\}$. A known result linking norm and spectral radio of a matrix A, and useful for us, is the following.

Theorem 1. *Let A be an $n \times n$ real valued matrix. Then the natural norm of A (say, given by the euclidean norm) verifies that $||A|| = \sqrt{\rho(A^t A)}$. If, further, A is symmetric then $||A|| = \rho(A)$.* □

2.3 The spectral clustering algorithm

Given data set $V = \{v_1, \ldots, v_n\}$ and a similarity relation $S = (s_{ij})_{1 \leq i,j \leq n}$ on V, the basic spectral clustering schema is the following

---

Input: $S = (s_{ij})_{1 \leq i,j \leq n}$ (similarity matrix),
k (number of clusters)
Output: Vector idx of k indices for the cluster sets
1. Compute the Laplacian matrix L associated to S
2. Compute the first k eigenvectors $u_1, \ldots, u_k$ of L
3. Let $U = (u_1 \ldots u_k)$, where u_i is a column in $\mathbb{R}^n$
4. For $i = 1, \ldots, n$, let y_i be the i-th row of U and
use k-means to clusters the rows $\{y_i : i = 1, \ldots, n\}$
of U as points in $\mathbb{R}^k$
5. For $j = 1, \ldots, k$, let $idx(j)$ be the index set
of all the rows of U labelled with j
(i.e. $i \in idx(j)$ iff y_i is in the j-th cluster)

When $L = L_{rw}$ we have essentially the algorithm proposed by Shi and Malik [10], and for $L = L_{sym}$, it is the algorithm of Ng, Jordan and Weiss [9].

2.4 What is going on?

It is worth reviewing a graphical point of view of clustering and its close relation with spectral clustering, in order to leave the reader in no doubt that spectral clustering will indeed partition the data as desired. We follow the exposition in [11] of this fact.

Given data set $V = \{v_1, \ldots, v_n\}$ and a similarity relation $S = (s_{ij})_{1 \leq i,j \leq n}$ on V, define the *similarity weighted graph* $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ and $v_i v_j \in E \iff s_{ij} > 0$, and the edge $v_i v_j$ has weight s_{ij}. Then the problem of clustering V becomes the *graph partition problem*:

> To find a partition of V such that edges between two different parts have low weight and edges within the same part have high weight.

The solution of this partition problem is equivalent to solving the *mincut problem*:

> To find a partition $A_1, \ldots, A_k$ of V which minimizes
>
> $$Cut(A_1, \ldots, A_k) = \sum_{i=1}^{k} cut(A_i, \overline{A_i})$$
>
> where $cut(A, \overline{A}) = \sum_{v_i \in A, v_j \notin A} s_{ij}$

A normalized form of the Cut function is the following:

$$RatioCut(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{cut(A_i, \overline{A_i})}{|A_i|}$$

Now we can see how Spectral Clustering relates to Graph Partition in the following way. (Below, $Tr(w)$ denotes the trace of w.) Given a partition A_1, $\ldots$, A_k of V, let

$$h_{ij} = \begin{cases} 1/\sqrt{|A_i|} & \text{if } \ v_i \in A_j \\ \\ 0 & \text{otherwise} \end{cases}$$

Define k indicator vectors $h_j = (h_{1j}, \ldots, h_{nj})^t$ and $H = (h_1, \ldots, h_k) \in \mathbb{R}^{n \times k}$.

We have

- $H^t H = I$
- For Laplacian $L = D - S$,

$$h_i^t L h_i = 2 \frac{cut(A_i, \overline{A_i})}{|A_i|}$$

Therefore

$$RatioCut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} h_i^t L h_i = \frac{1}{2} \sum_{i=1}^{k} (H^t L H)_{ii} = \frac{1}{2} Tr(H^t L H)$$

Then minimizing $RatioCut(A_1, \ldots, A_k)$ is equivalent to

$$\min_{A_1, \ldots, A_k} Tr(H^t L H) \ \text{ for } \ H = (h_1, \ldots, h_k)$$

This is an NP-hard optimization problem. Thus, one relaxes H to real values, and the problem becomes

$$\min_{H \in \mathbb{R}^{n \times k}} Tr(H^t L H) \ \text{ subject to } H^t H = I$$

The solution is obtained by choosing H as matrix with columns the first k eigenvectors of L.

3 Practical considerations

3.1 Solving an eigensystem

An essential feature in the spectral clustering algorithm (sec. 2.3) is the need to compute the eigenvalues and eigenvectors of the Laplacians. For large sparse matrices, as those describing a large web site, a straightforward resolution is computationally forbidding, so we need some numerical method to compute approximate solutions to these eigensystems.

A simple but effective iterative method for computing the largest (in absolute value) eigenvalue is the Power Method. Given an $n \times n$ matrix A, with eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, the following schema, known as the Power Method, computes successive approximations to the dominant eigenvector:

```
Input: arbitrary n dimensional vector x_0, and matrix A
for k = 1, ..., m do
    y_k = A x_{k-1}
    x_k = y_k / ||y_k||
end for
```

If the true dominant eigenvector is u_n then the associated eigenvalue $\lambda_n = \frac{\langle u_n, Au_n \rangle}{\langle u_n, u_n \rangle}$. Thus we can approximate the value of λ_n by the vector computed in the previous algorithm with what is known as the *Rayleigh quotient*:

$$\lambda_n \approx \frac{\langle x_m, Ax_m \rangle}{\langle x_m, x_m \rangle}.$$

Although the Power Method is considered obsolete by numerical analysts, it is extremely practical for solving spectral problems with very high usage of memory. In fact, it is the method employed by Google's PageRank algorithm (see [5]). For computing the k first eigenvectors there are precise and fast methods based on projections over Krylov subspaces [13]. Within this family of spectral methods we chose to use the method of Arnoldi, which we briefly explain below.

The Krylov subspace methods build up Krylov subspaces

$$K_j(A, x) = span\{x, Ax, A^2x, \ldots, A^{j-1}x\}$$

and look for good approximations to eigenvectors. In general, the vectors $A^i x$, as i increases, have their direction pointing towards the dominant eigenvector, which produces, more often than not, a basis for an ill-posed system (in a sense explained below in section 3.3). The Arnoldi method avoids this problem by building an orthonormal basis by means of a Gram-Schmidt process.

In general, given an $n \times n$ complex matrix A and a random complex vector x of dimension n, the Arnoldi process begins with the normal vector $u_1 = x/||x||$ and continues recursively building orthonormal vectors. Having produced $u_1, \ldots, u_j$, it produces u_{j+1} as follows:

$$h_{ij} = \langle Au_j, u_j \rangle = u_i^* Au_j, \quad i = 1, \ldots, j, \tag{2}$$

where $\langle x, y \rangle = y^* x = \sum_{j=1}^{n} x_j \bar{y_j}$ is the standart complex inner product.

Then one constructs the vector

$$\hat{u}_{j+1} = Au_j - \sum_{i=1}^{j} u_i h_{ij} \tag{3}$$

and, if $h_{j+1,j} = ||\hat{u}_{j+1}|| \neq 0$ we make

$$u_{j+1} = \frac{\hat{u}_{j+1}}{h_{j+1,j}}. \tag{4}$$

If the coefficients $h_{1j}, \ldots, h_{jj}$ are defined as shown in (2), then the new vector $\hat{u}_{j+1}$ is orthogonal to $u_1, \ldots, u_j$ and so u_{j+1} is orthonormal. If $Au_j \notin span\{u_1, \ldots, u_j\}$, then certainly $\hat{u}_{j+1} \neq 0$. Conversely, if $Au_j \in span\{u_1, \ldots, u_j\}$, the choice of coefficients, given by (2), forces $\hat{u}_{j+1} = 0$. In this case $span\{u_1, \ldots, u_j\} = K_j(A, x)$ is invariant under A, and the process terminates.

Suppose that, after k steps of the Arnoldi process, we have generated orthonormal vectors $u_1, \ldots, u_k$ linearly independent. Let U_k denote the $n \times k$ matrix whose columns are $u_1, \ldots, u_k$ and let $H_k = (h_{ij})$ denote the $k \times k$ upper Hessenberg matrix given by (2). Then, if $\mathcal{K}_k(A, u_1)$ is invariant under A, the equations (3) and (4) imply $AU_j = \sum_{i=1}^{j+1} u_i h_{ij}$ and so $AU_k = U_k H_k$ or $H_k = U_k^* A U_k$, due to the orthonormality of U_k. Normally $k << n$ and the computation of the eigenvectors of H_k is very easy (and with low cost). These eigenvectors are good approximations of the k first eigenvectors of A. The approximation improves for eigenvectors associated with eigenvalues on the periphery of the spectrum.

An advantage of using the method of Arnoldi for solving eigensystems is that it is implemented in MatLab under the function *eigs* (cf. the manual [7]). The function *eigs* is based on the software package ARPACK written in Fortran (see [6]).

3.2 Storing a sparse matrix

Our preferred format for representing a sparse matrix for our algorithms is the *Compressed Row Storage* (CRS). This is a data structure convenient for sparse matrices due to its minimal storage requirements. A CRS structure is composed of three vectors (usually implemented with pointers in C++):

val is the vector containing the non null elements of the matrix;
col_ptr contains the column indices of elements in **val**,
row_ptr contains the indices of the elements of **val** that start in a new row.

MatLab uses its own CRS format for storing sparse matrices. For each non null entry of the matrix, it stores a tuple $((x, y), val)$ where (x, y) describes the position of the element in the matrix, and val its value.

WIRE, our selected web crawler (which will be briefly described in 3.4), does not have the feature of outputting the collection of links that it retrieves from the web, as an adjacency matrix in CRS format. We wrote that functionality so that we could pass the data from WIRE to the MatLab's *eigs* function.

3.3 On the conditioning number of a laplacian

Computing the eigenvectors of a Laplacian is the key feature in a spectral clustering algorithm. We have pointed out the need to use numerical methods to solve approximately this problem, in part due to the size of the input matrices. These numerical methods are implemented in real computers with limitations on the representation of numbers, floating point precision, and so forth. The worse the condition of the numerical problem, the more the converge of the methods will be affected by those physical limitations. Informally a numerical problem (e.g. a system of linear equations) is badly conditioned when its solution is prone to big changes under small perturbations of the initial data. To know how ill-conditioned is a numerical problem, one computes the *condition number* of its associated matrix.

Definition 1 (Condition Number). *The Condition Number of a regular matrix A is the number*

$$\mu(A) = ||A||||A^{-1}||$$

Since

$$1 = ||I|| = ||AA^{-1}|| \leq ||A||||A^{-1}|| = \mu(A)$$

the optimal condition number of a regular matrix is attained at 1. Given A, a symmetric and regular matrix with spectrum $\sigma(A)$, and given that $\sigma(A^{-1}) = \{1/\lambda : \lambda \in \sigma(A)\}$, using Theorem 1 we have that

$$\mu(A) = \frac{\max\{|\lambda| : \lambda \in \sigma(A)\}}{\min\{|\lambda| : \lambda \in \sigma(A)\}} \tag{5}$$

From this equation we can interpret the condition number as the rate of change between the extreme deformations produced by A, seen as a matrix of a linear transformation.

Now it should be clear from equation (5) and the fact that Laplacian matrices have at least one zero eigenvalue (Proposition 1), that the system $Lv = \lambda v$ (i.e. an eigensystem for Laplacian L) is ill-conditioned, and so it is expected that problems of convergence will arise in the numerical method of Arnoldi.

Indeed, in our first straightforward implementation of the spectral algorithm (as in §2.3) working with the adjacency matrix of a large web graph (i.e. that of a university, whose dimension is of the order of 10^5 nodes), the function *eigs* would quickly advert of the bad conditioning of the associated Laplacian matrix, therefore aborting the computation of eigenvectors.

We proposed, as a fix to overcome this computational difficulty, to work with a perturbed version of the Laplacian matrix L. We shall show below that this does not affect the final result.

Definition 2. *Let L be a Laplacian matrix. The* perturbed matrix *associated to L is*

$$L_t = L + tI \qquad \textit{with } t \in \mathbb{R}^+$$

where I is the identity matrix.

The spectrum of the perturbed matrix is $\sigma(L_t) = \sigma(L+tI) = \{\lambda_1+t, \ldots, \lambda_n+t\}$, and therefore, from equation (5), we get

$$\mu(L_t) = \frac{\max\{|\lambda + t| : \lambda \in \sigma(L)\}}{\min\{|\lambda + t| : \lambda \in \sigma(L)\}} = \frac{\lambda_n + t}{\lambda_1 + t} = \frac{\lambda_n}{t} + 1 \tag{6}$$

From this result it follows that if we add the spectral radio $\rho(L)$ to the elements on the diagonal of L, we will obtain a condition number equal to 2, and if we add a number greater than $\rho(L)$, the condition number will be closer to 1. This means that we have to shift the spectrum of L, and it is important to know that this shifting does not alter the eigenvectors of L, as we show next.

Theorem 2. *The eigenvectors of L associated to the eigenvalue λ are the same as the eigenvectors of L_t associated to the eigenvalue $\lambda + t$.*

Proof. If u is an eigenvector associated to the eigenvalue λ of L, we have $Lu = \lambda u$, and in consequence $L_t u = (L+tI)u = Lu+tIu = \lambda u+tu = (\lambda+t)u$ □

The previous result applies to the unnormalized matrix L as well as to the matrix L_{sym}, hence insuring a good conditioning for the matrices needed in the spectral clustering algorithm.

3.4 Web crawler

WIRE is a powerful open source webcrawler designed by Carlos Castillo as part of his PhD Thesis [3,12]. It is oriented to information extraction and making of web statistics, and as opposed to the classical architecture of crawlers based on two modules, a planner and an administrator, it separates the tasks among four specialized and highly efficient modules: the *manager*, the *harvester*, the *gatherer*, and the *seeder*. Among its many features we were particularly interested in two: 1) allows planning in short as well as long term (e.g. the crawler can be stopped and later restart at last visited place, keeping history); 2) parametrize by a unique XML script (`wire.conf`).

Now, no webcrawler generates the adjacency matrix of the graph that it is visiting, and WIRE is no exception. However, thanks to its open code policy and documentation, we could modify the utility `wire-info-extract`, which is the tool available for consulting the repositories of visited web pages, and add the functionality `wire-info-extract -matrix` to get the adjacency matrix of the pages retrieved in CRS format.

4 Spectral clustering algorithm in practice

We now present our modified version of the spectral clustering algorithm which works well in practice, since it includes all the fixes to the computational problems discussed in the previous section. In particular: 1) the Laplacian matrix is perturbed (shifted appropriately) so that the eigensystem is well conditioned; 2) the k-means algorithm is repeated a finite number of times to avoid convergence to local optimal of the function that measures the intra-cluster similarity (we have determined that 10 repetitions is sufficient in practice).

The algorithm works with all three forms of Laplacians discussed in this paper (L, L_{sym} and L_{rw}); although in our pseudo-code presentation we refer only to L_{rw}, and in this sense it is a normalized spectral clustering. Also we include some of the actual MatLab commands used by the true program.

Modified normalized spectral clustering

Input: $S = (s_{ij})_{1 \le i,j \le n}$ (similarity matrix),
k (number of clusters)
Output: Vector idx of k indices for the cluster sets
1. Compute the Laplacian matrix L_{rw} associated to S:
$D = diag((sum(S, 2)).\hat{}\,(-1))$;
$L_{rw} = D * S$
2. Shift the spectrum of L_{rw} to enhance its condition number by adding the spectral radio to the main diagonal:
$major = eigs(L_{rw}, 1)$;
$L_{rwt} = L_{rw} + (major * speye(size(L_{rw})))$;
3. Compute the first k eigenvectors $u_1, \ldots, u_k$ of L_{rw}
Let $U = (u_1 \ldots u_k)$, where u_i is a column in $\mathbb{R}^n$:
$[U, lambda, flag] = eigs(L_{rwt}, k, `sm`, options)$;
4. For $i = 1, \ldots, n$, let y_i be the i-th row of U and use k-means to cluster the rows $\{y_i : i = 1, \ldots, n\}$ of U as points in $\mathbb{R}^k$. The procedure is repeated 10 times to avoid local optimal points
$idx = kmeans(U, k, `replicates`, 10)$
5. For $j = 1, \ldots, k$, let $idx(j)$ be the index set of all the rows of U labelled with j
(i.e. $i \in idx(j)$ iff y_i is in the j-th cluster)

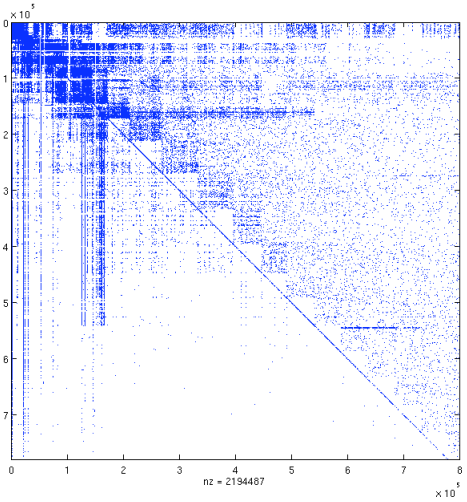
---

5 Experimental Results

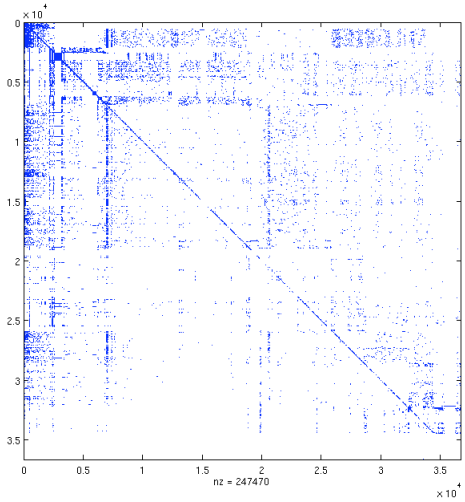
We applied the modified spectral clustering algorithm to partition large internet domains. Our selected samples were universities domains (uab.es, uva.es) and electronic encyclopaedia (enciclopedia.us.es). These are websites of about 10^5 non isolated nodes.

The similarity relation $s_{ij} = s(v_i, v_j)$ between two web pages was defined in terms of their neighbourhoods as follows. For a web page v_i, let V_i be the set of pages that have a link to or from v_i. Then $s_{ij} = jac(V_i, V_j)$; that is, pages v_i and v_j are similar if the jaccard measure (equation (1)) between their respective neighbours, V_i and V_j, is greater than some positive threshold. This similarity measure models a relation of local vicinity among web pages, based on common linkage. We determined that $k = 6$ was a reasonable number of clusters. The algorithm performed correctly and terminated in a reasonable time, which depended on the size of the input matrix.

We present below two plots representing the adjacency matrices of the domains crawled by WIRE, so the reader has an idea of the size and distribution of the input.

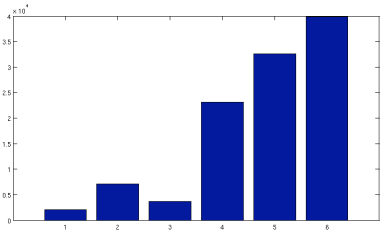


Adjacency matrix of uva.es

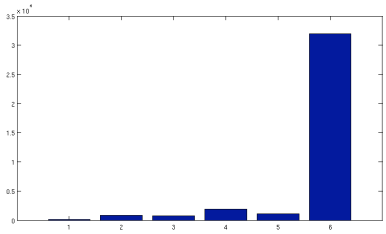


Adjacency matrix of uab.es

The following plots present a comparison of volume (the sum of similarities with respect to the centroid of the cluster) with the size (number of elements) of each of the six clusters found. The reader should not be surprise by the apparent lack of homogeneity of volume versus size on the six clusters. Recall from the graphical interpretation given in §2.4, that the goal of spectral clustering is equivalent to minimising the function $RatioCut(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{cut(A_i, \overline{A_i})}{|A_i|}$. Thus there are two objectives, and possibly independent to each other, namely, to minimise the connectivity among clusters (the cut), which should increase the inter-connectivity (hence volume), while balancing the sizes.



volume of clusters



number of elements per cluster

References

[1] Arratia, A., Marijuán, C. (2006). Cómo mejorar el PageRank de un árbol, in E. M. Moro et al (eds.): V Jornadas de Matemática Discreta y Algoritmica (Pub. Univ. Valladolid, 53-60).

[2] Arratia, A., Marijuán, C. (2009). Ranking pages and the topology of the web. (submitted).

[3] Castillo, C. (2004). Effective Web Crawling (Ph. D. Thesis in Computer Science, University of Chile).

[4] Chung, F. (1997). *Spectral graph theory*. Washington: Conference board of the Mathematical Sciences.

[5] Langville,A.N., Meyer, C.D. (2006). Google's PageRank and Beyond: The Science of Search Engine Rankings. Princeton University Press.

[6] Lehoucq, R. B., Sorensen, D. C., Yang, C. (1998). ARPACK User's Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods. SIAM Publications, Philadelphia.

[7] Matlab 7 Mathematics (2008). MathWorks, Inc. Available from http://www.mathworks.com/access/helpdesk/helpdesk.html

[8] Mohar, B., Alavi, E. Y., Chartrand, G., Oellermann. O. R., Schwenk, A. J. (1991). The laplacian spectrum of graphs. In *Graph Theory, Combinatorics, and Applications*, vol.2, 871-898. Wiley.

[9] Ng, A., Jordan, M., and Weiss, Y. (2002). On spectral clustering: analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems 14. MIT Press.

[10] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), 888-905.

[11] von Luxburg, U. (2007). A Tutorial on Spectral Clustering. Statistics and Computing 17(4): 395-416. (A previous version of this paper appeared as Technical Report 149, Max Planck Institute for Biological Cybernetics, 2006.)

[12] WIRE, http://www.cwr.cl/projects/WIRE/

[13] Watkins, D. S. (2007). The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods. Siam.