Technical Report
Spectral Clustering of Large Advertiser Datasets
Part I

Leonid Zhukov
Overture R&D

April 10, 2003
Revised: January 15, 2004

Contents

1. Introduction

A pervasive problem in data mining is finding natural grouping and hidden trends in data. In this report we consider a pay-per-performance search listing dataset, which consists of a set of advertisers and a set of keywords those advertisers are bidding on. Due to the competitive nature of the market each keyword may be bidded on by many advertisers and almost every advertiser bids on more than one keyword.

The "advertiser-keyword" relationship can be represented by a bipartite graph with edges connecting advertisers on one side of a graph with keywords they are bidding on the other side of the graph. It is natural that advertisers with common interests bid on the same subset of keywords and thus form a submarket, i.e. a collection of advertisers and bidded keywords that are connected stronger to each other than to the rest of the dataset. From a graph theoretical point of view, the submarket definition is equivalent to the definition of approximate cliques, while from a data mining perspective it is the definition of clusters. Thus the problem of finding submarkets can be approached through clustering or graph partitioning of a bipartite graph [2], [5], [27].

In this study we use Overture's US "advertiser-keyword" dataset with over 145,000 advertisers, 127,000 bidded keywords and more than 3.1 million bids and its small densely connected subset of 3,000 bidded keywords, 2,000 advertisers and 100,000 bids.

1.1 Vector space model

An alternative to bipartite graph representation for the data can be given by an "advertiser-keyword" matrix whose rows corresponds to advertisers, columns to bidded keywords and nonzero matrix elements to exiting bids. The number of columns in this matrix is equal to the number of unique bidded keywords and the number of rows is equal to the number of unique advertisers active on the market. This matrix is sparse, since the majority of the advertisers bid only on a small number of terms. Thus every column of this matrix represents an "advertiser vector" described in the "bidded keyword space" and every raw is a "bidded keyword vector" in the "advertiser space". This constitutes a vector space model [1].

1.2 Clustering

Clustering is a well studied problem in data mining there are exist multiple clustering algorithms [3, 10, 23, 9, 16] In general, one can distinguish two type of clusterings - hierarchical clustering and flat clusters. The former methods provides a data hierarchy and can be converted to flat clusters by "cutting" the tree on a certain level. By construction, hierarchical clustering can be *agglomerative* or "bottom up" and *divisive* partitioning "top down" . Agglomerative clustering typically examines every item in the data and assembles clusters by grouping items together starting from singletons. Divisive approach, on the opposite, starts by looking at the entire dataset and finding the way to split it into several large group. Both approaches are recursive and build a hierarchical structure. Agglomerative approach is inherently $O(N^2)$, partitioning could be done in $O(NlogN)$. Partitioning usually provides better large clusters, on the top of the hierarchy, agglomerative is best at the bottom of hierarchy with small groups. Flat clustering methods usually require a priori knowledge of the number of clusters and based on iterative assigning elements to the clusters (K-means).

In this report we will consider one of the divisive type of clustering algorithms based on spectral properties of the data matrix. The algorithms will consider data as a graph and clustering problem will be formulated as a graph partitioning.

When formulating a clustering problem for a given data, one should decide what are data points (objects) and what constitutes attributes. A reasonable starting point for us is to cluster advertisers using the distribution of bidded terms. That is, advertisers are data objects and bidded terms are attributes (coordinates). Then advertisers whose choice of bidded terms significantly overlap belongs to the same cluster.

So we need to define a similarity metric between advertiser (objects) based on the bidded terms (attributes). Mathematically, in the vector space model, this similarity can be expressed through the *cosine* of the angle between data

vectors - columns of the term-advertiser matrix:

$$C_a(a_1, a_2) = cos(a_1, a_2) = \frac{a_1 \cdot a_2}{\|a_1\| \cdot \|a_2\|} \tag{1}$$

Thus, advertisers which bid on the identical set of terms, independently of the size of that set, will always have the largest correlation $C_a = 1$. "Orthogonal" advertisers do not have any terms in common and $C_a = 0$

Analogously, one can try to cluster bidded terms, considering them as vectors in advertiser space (rows of matrix) and using corresponding co-occurrence among advertisers. Two bidded terms have high correlation value and can belong to the same cluster if the same set of advertisers bids on them and zero if there is is no overlap.

$$C_p(p_1, p_2) = cos(p_1, p_2) = \frac{p_1 \cdot p_2}{\|p_1\| \cdot \|p_2\|} \tag{2}$$

Naturally, one might consider clustering based on both metrics, i.e, simultaneous clustering of terms and advertisers. In our work we will be using the last approach.

2. Methods

2.1 Basic graph terminology

A graph $G = (V, E)$ consists of a set of vertices $V = \{V_1, V_2...\}$ and a set of edges with edge weight e_{ij} connecting those vertices. Below we consider only undirected graphs.

A graph can be represented using an adjacency matrix

$$\mathbf{M} = \begin{cases} e_{ij}, & if\ \exists\ \{i, j\} \\ 0, & otherwise \end{cases} \tag{3}$$

Since our graphs are undirected, the adjacency matrix is always symmetric.

The vertex set of a graph can be divided into two groups according to some properties, thus inducing a graph partitioning. The degree of dissimilarity between the two subgraphs can be computed as a total weight (sum of all weights) of the edges that have been removed during the *cut* :

$$cut(V_1, V_2) = \sum_{e_{ij}} e_{ij} = \sum_{i \in V_1, j \in V_2} e_{ij} \tag{4}$$

Classical graph partition algorithms try to minimize the cut value, i.e., find such partitions V_1 and V_2 that minimize the objective function

$$Q(V_1, V_2) = cut(V_1, V_2); \tag{5}$$

2.2 Graph partitioning for clustering

When using graph partition for clustering, we usually value big clusters more than small ones and would like to prevent cutting off singletons or very small subgraphs. This can be achieved by augmenting optimization function with a normalization procedure. The new objective function can be written as

$$Q(V_1, V_2) = \frac{cut(V_1, V_2)}{W(V_1)} + \frac{cut(V_1, V_2)}{W(V_2)} \tag{6}$$

where W is the sum of the weights of all nodes in the partition

$$W(V) = \sum_{i \in V} w_i \tag{7}$$

Various choices of node weight function W lead to different partition criteria and clustering results. For example,

- Ratio-cut [18]. All vertices are given the same weight $w_i = 1$ and $W(V_1) = \|V_1\|$ is the number of vertices in partition. Then

$$Q(V_1, V_2) = \frac{cut(V_1, V_2)}{\|V_1\|} + \frac{cut(V_1, V_2)}{\|V_2\|} \tag{8}$$

- Normalized cuts [25]. Weight of every vertex is equal to the the sum of the weights of incident edges, $w(i) = \sum_k E_{ik}$. Then

$$Q(V_1, V_2) = \frac{cut(V_1, V_2)}{\sum_{i \in V_1, k \in V} E_{i,k}} + \frac{cut(V_1, V_2)}{\sum_{i \in V_2, \in V} E_{i,k}} \tag{9}$$

  Further discussion of the spectral graph partitioning method for clustering can be found in [6, 26, 17, 22, 7]

2.3 Vector space model connection

It is interesting to define the optimization -cut criteria from the vector model perspective. It could be easily done for single side graph with nodes corresponding to advertisers and edges carrying weight according to the similarity metric between advertisers, in our case, the *cos* of angle between advertiser vectors. It is convenient to define a centroid vector for a cluster:

$$C_V = \frac{1}{N} \sum_{i \in V} d_i \tag{10}$$

Then the cut value can be expressed as

$$\begin{aligned} cut(V_1, V_2) = \sum_{E_{ij}} E_{ij} = \sum_{i \in V_1, j \in V_2} cos(d_i, d_j) = \\ \sum_{i \in V_1, j \in V_2} \frac{d_i \cdot d_j}{\|d_i\| \, \|d_j\|} = \sum_{i \in V_1} \frac{d_i}{\|d_i\|} \cdot \sum_{j \in V_2} \frac{d_j}{\|d_j\|} = \\ N_1 N_2 (C_1 \cdot C_2) = (N_1 N_2)^2 cos(C_1, C_2) \end{aligned} \tag{11}$$

which is a scaled cosine distance between the centroids of the two clusters.

The ratio cut (8) criterion is then equivalent to

$$Q(V_1, V_2) = \frac{N_1 N_2 (C_1 \cdot C_2)}{N_1} + \frac{N_1 N_2 (C_1 \cdot C_2)}{N_2} = N(C_1 \cdot C_2) \tag{12}$$

Thus, minimization of the ratio-cut value corresponds to clustering advertiser in such groups, that minimizes the cosine similarity between their centroids.

The normalized cuts (9) can also be expressed in the vector model:

$$Q(V_1, V_2) = \frac{N_1 N_2 (C_1 \cdot C_2)}{N_1 N (C_1 \cdot C)} + \frac{N_1 N_2 (C_1 \cdot C_2)}{N_2 N (C_2 \cdot C)} = \tag{13}$$

$$\frac{N_2}{N} \frac{(C_1 \cdot C_2)}{(C_1 \cdot C)} + \frac{N_1}{N} \frac{(C_1 \cdot C_2)}{(C_2 \cdot C)} \tag{14}$$

Thus, normalized cuts optimizes the sum of the ratios of cosine similarity between the clusters over the cosine similarity between the cluster centroid and the entire dataset centroid ("average" advertiser).

2.4 Spectral graph partitioning

Spectral graph partitioning problem can be formulated as a global minimization of the objective function through the solution of an eigenvalue problem. The second smallest eigenvector provides the lower bound on the value of optimization function and the second eigenvector (Fiedler vector [19]) is used to construct the partitioning of the graph. The intuition for spectral methods comes from related physical problem, where we consider a graph as a mesh of springs connecting masses and look for oscillating modes (standing waves) of the system. The second eigenvalue will describe the second harmonic of oscillations, which, in fact, divides the mesh into two parts. (Imagine a string fixed at both ends: the second eigenvalue/vector corresponds to the second harmonic which has a node in the middle of the string and one side is displaced up and the other down). Applications of Laplace eigenvalues to graph partitioning have been extensively studied in literature [20, 21, 14, 12, 13, 15].

Consider partitioning of a graph in two parts and define a partitioning indicator vector $p = [-1, .., 1]$ of ± 1 such that for all nodes belonging to one partition $p_i = -1$ and to the other $p_i = 1$. then Eq. 4 can be expanded and simplified to

$$\begin{aligned} cut(V_1, V_2) = \sum_{E_{ij}} E_{ij} = \frac{1}{4} \sum_{i>j} E_{ij} (p_i - p_j)^2 = \frac{1}{8} \sum_{i,j} E_{ij} (p_i - p_j)^2 \\ = \frac{1}{8} (\sum_{i,j} E_{ij} p_i^2 - 2 \sum_{i,j} E_{ij} p_i p_j + \sum_{i,j} E_{ij} p_j^2) \\ = \frac{1}{4} (\sum_i p_i^2 \sum_j E_{ij} + \sum_{i,j} E_{ij} p_i p_j) = \frac{1}{4} (\sum_i p_i^2 D_{ii} - \sum_{i,j} E_{ij} p_i p_j) \end{aligned} \tag{15}$$

where

$$D_{ii} = \sum_j E_{ij}. \tag{16}$$

For non-weighted graphs $E_{ij} = 1$ and D is a degree matrix, i.e. every diagonal element is equal to a degree of the corresponding vertex in the graph - number of incident edges on that node.

In vector notation Eq. (15) becomes

$$cut(V_1, V_2) = \frac{1}{4}(p^T D p - p^T E p) = \frac{1}{4} p^T (D - E) p = \frac{p^T L p}{4} \tag{17}$$

where we introduced a Laplacian matrix of the graph

$$\mathbf{L} = \mathbf{D} - \mathbf{E} \tag{18}$$

Laplacian matrix

Laplacian matrix of a graph is an adjacency matrix with diagonal elements equal to the negative sum of the rest of the matrix elements from the same row. Thus total sum of the elements in a row of Laplacian matrix is equal to zero. Formally, Laplacian matrix can be defined as

$$\mathbf{L} = \begin{cases} -E_{ij}, & if\ \exists\ \{i,j\},\ i \neq j \\ \sum_k E_{ik}, & if\ \ i = j \\ 0, & otherwise \end{cases} \tag{19}$$

The Laplacian matrix is a symmetric positive semi-definite matrix with real non-negative eigenvalues and a complete set of orthogonal eigenvectors. By construction, matrix elements in every raw of L sums up to 0 and, thus, $e_1 = [1, 1, ...1]$ is an eigenvector of L with corresponding zero eigenvalue, $Le_1 = 0$. Since matrix is positive semi-definite, $\lambda_1 = 0$ is its smallest eigenvalue.

An important theorem due to Fiedler [19] states that the number of connected components in a graph G is equal to number of zero eigenvalues. If there is only one connected component, the smallest eigenvalue is 0 and the second smallest eigenvalue is above zero.

Spectral Methods

Classical graph partitioning spectral method minimizes the objective function Eq.(5). Combining the equations Eq.(5) and Eq.(17) we obtain

$$Q(V_1, V_2) = cut(V_1, V_2) = \frac{p^T L p}{4} \tag{20}$$

Thus minimization of the function or finding the smallest cut becomes a problem of optimizing the value of Eq.(20) with respect to integer-valued ± 1 vector p.

The conversion of partitioning problem into integer non-linear optimization problem does not make it easier, since finding the global minimum of the function Eq.(20) and the best p requires the exhaustive search over all possible

choices of distribution of -1 and 1 in the vector p and this is factorial hard (NP-hard) problem. But if we relax the problem, by allowing vector elements to take any real value between -1 and 1, the problem will be moved from a discrete to continues domain and there exist efficient method for optimization of continues functions! It is important to understand though, that we will be looking for an *approximate solution* to an integer optimization problem. The original problem, formulated in discrete space have a solution vector existing only in the corners of a unit hypercube $\{\pm 1, \pm 1, ... \pm 1\}$. The continues solution must be constructed in a such a way, that it also goes through the corners of hypercube since all possible integer solutions *must be* part of the continues set. This requirements can be satisfied by enforcing normalization of the continues solution vector. Since $p^T p = N$, total number of nodes in the graph, we enforce $\|x\| = x^T x = N$. This equation means that the solution points lie on a hypersphere centered at the origin and going through the corners of the unit hypercube.

Thus we have replaced an integer discrete optimization problem Eq.(20) with continues optimization under the constraint

$$Q(x) = \frac{x^T L x}{4}, \quad \|x\| = N. \tag{21}$$

or, using Lagrange multipliers,

$$Q(x) = \frac{x^T L x}{4} - \hat{\lambda}(x^T x - N) \tag{22}$$

Since Eq.(22) is a quadratic form, it can be easily minimized by diagonalization. Variational solution to this optimization problem is given by the solution of eigensystem:

$$Lx = \lambda x \tag{23}$$

and minimum of $Q(x)$ is the Rayleigh Quotient reached on $x = e_1$

$$\min_x \rho(x, L) = \min_x \frac{x^T L x}{x^T x} = \lambda_1 \tag{24}$$

Since $e_1 = [1, ..1]$ is an eigenvector of L with zero eigenvalue, which *does not* provide a partitioning, we enforce another constraint on the solution: $x \perp e_1$. Then the minimum is achieved on the second eigenvector (this is a special case of Courant Fischer minimax theorem)

$$\min_{x, x \perp e_1} \frac{x^T L x}{x^T x} = \lambda_2 \tag{25}$$

As follows from the previous discussion to approximate a discrete solution from the continues we have to round eigenvector elements to ± 1 and thus split nodes into two sets. One of the possible schemes is to separate elements according their sign into positive and negative groups, thus converting x into p using $p = sign(x)$. If balanced partition is desired, we could use median value of the

vector elements as a separator. One can also use grouping of the elements to to find *natural cut* boundaries, which corresponds to clustering.

The results of spectral computations could be used for more than just split-partitioning the graph. Equation (21) can be expanded back to

$$Q(x) = \frac{x^T L x}{4} = \sum_{ij} E_{ij}(x_i - x_j)^2 \tag{26}$$

The minimum of this expression is reached when the neighboring elements of vector x have the smallest difference, thus enforcing global ordering on the minimizing configuration of nodes. In practice this ordering can be achieved by sorting elements of the the second eigenvector (in ascending order, for example) and using indexes into the elements as new indexes for the matrix rows and columns. In this case various partitions can be achieved by moving separator along the vector. The optimal position can be determined by heuristics, using median values to get more balanced cut or explicitly compare some additional cut values for all bi-partitions. It is possible to do exhaustive comparisons in this case because there are only $N-1$ non-empty partitions can be configured - nodes are already sorted!. Thus spectral method essentially reduced $O(N!)$ hard problem to an approximate problem, but with order $O(N)$ complexity.

One can also apply various standard low-dimensional clustering techniques (K-means, for example) to the eigenvector elements to group them into several clusters and thus partition the graph into more than two parts. We will explore this option later.

Generalized Spectral method

One can show [5], that optimizing the function $Q(V_1, V_2)$ from Eq. (6) is equivalent to optimizing the following *weighted* Rayleigh quotient

$$Q(V_1, V_2) = \frac{cut(V_1, V_2)}{W(V_1)} + \frac{cut(V_1, V_2)}{W(V_2)} = \frac{q^T L q}{q^T W q} \tag{27}$$

where elements of vector q take values $q_i = \{+\sqrt{W(V_2)/W(V_1)}, -\sqrt{W(V_1)/W(V_2)}\}$.

Its minimum under the *weighted* constraint $x \perp We_1$ value is achieved on the second eigenvector

$$\min_{x, x \perp W e_1} \frac{x^T L x}{x^T W x} = \lambda_2 \tag{28}$$

of the generalized eigenproblem

$$Lx = \lambda W x. \tag{29}$$

Spectral method for bipartite graphs

Let M be a bipartite graph corresponding to a term-advertiser matrix A with m terms and n advertisers (in our case $m >> n$)

$$\mathbf{M} = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \tag{30}$$

In this ordering, the first m nodes in the graph contain terms and the last n nodes - advertisers.

To partition this graph using spectral methods, we first formulate a generalized eigenvalue problem that corresponds to global optimization of weighted function Eq. (29). We use "normalized cuts" (see Eq. (9)) approach for weighting function.

$$Lz = \lambda W z \tag{31}$$

The Laplacian matrix for this equation is formed according to definition Eq.(18)

$$\mathbf{L} = \begin{pmatrix} D_1 & -A \\ -A^T & D_2 \end{pmatrix} \tag{32}$$

and

$$\mathbf{W} = \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix} \tag{33}$$

where $D_{1,ii} = \sum_j A_{ij}$ and $D_{2,jj} = \sum_i A_{ij}$ are diagonal matrices with diagonals equal to correspondingly sums of rows and columns of matrix A. The solution vector can be thought of as a sequence of m nodes (sub-vector x) corresponding to bidded terms followed by n nodes (sub-vector y) corresponding to advertisers $z = [x, y]'$.

Written out component-wise the eigensystem becomes:

$$\begin{pmatrix} D_1 & -A \\ -A^T & D_2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{34}$$

This matrix eigensystem could be solved directly, but its dimension $(m+n)$ x $(m+n)$ makes such computations expensive. We will transform this system into equivalent one, but with mxn size.

We start by introducing row and column normalization for data matrix $\mathbf{A}$:

$$A_n = D_1^{-1/2} A D_2^{-1/2} \tag{35}$$

Substituting the expression of A through A_n into the system we get

$$\begin{aligned} D_1 x - D_1^{1/2} A_n D_2^{1/2} y &= \lambda D_1 x \\ -D_2^{1/2} A_n^T D_1^{1/2} x + D_2 y &= \lambda D_2 y \end{aligned} \tag{36}$$

Since D_1 and D_2 are non-singular,

$$\begin{aligned} D_1^{1/2} x - A_n D_2^{1/2} y &= \lambda D_1^{1/2} x \\ -A_n^T D_1^{1/2} x + D_2^{1/2} y &= \lambda D_2^{1/2} y \end{aligned} \tag{37}$$

After introducing new variables $u = D_1^{1/2} x$ and $v = D_2^{1/2} y$ the system can be expressed as

$$\begin{aligned} u - A_n v &= \lambda u \\ -A_n^T u + v &= \lambda v \end{aligned} \tag{38}$$

and simplified to

$$A_n v = (1-\lambda)u \qquad (39)$$
$$A_n^T u = (1-\lambda)v$$

These equations describes SVD decomposition of matrix A with singular values

$$\sigma = 1 - \lambda \qquad (40)$$

$$A_n = u\sigma v^T \qquad (41)$$

and can be computed by solving, for example, for v

$$(A_n^T A_n)v = \sigma^2 v \qquad (42)$$

and then recovering u by

$$A_n v = \sigma u \qquad (43)$$

We prefer to perform computations using $(A_n^T A_n)$ instead of $(A_n^T A_n)$ since this matrix is nxn and in our case $n << m$.

Note that because of Eq.(40) we are looking for the second *largest* eigenvalue and eigenvector.

Hierarchical spectral clustering

We can easily devise a recursive procedure using spectral method as a partitioning technique on every level. For every subgraph we extract corresponding adjacency matrix from the graph adjacency matrix and after re-numbering apply spectral bipartition again. Thus we build a binary tree with leaves containing nodes. Every leaf will ultimately contain at least one term-advertiser pair, but might also have several terms corresponding to one advertiser (if that is the tightest cluster) and vise versa. We devise several stopping criteria for the tree construction.

Some further ideas for the recursive spectral clustering can be found [4].

3. Implementation

The term-advertiser matrices are very large and sparse and since we are interested only in the second eigenvector, we use Lanczos iterative approach to numerically solve Eq. (42). The actual implementation is provided in the Arpack^{++} library [24, 11]. We use Arpack reverse communication interface providing the results of matrix vector multiply on every iteration. This product is computed using an intermediate vector for the storage and consequently creates $A_n v = z$; $A_n z = v'$. Thus we perform two sparse matrix vector multiply on every iteration instead of computing combined matrix. In recursive spectral clustering, when the smallest dimension of the matrix becomes less than $m, n < 20$, we switch to solving Eq. (34) instead of Eq. (42) and use direct

diagonalization methods from LAPACK [8] library . In practice such combined approach increases overall performance of the system.

Details of the scalable implementation and software library description will be presented in Part II of this report.

4. Experimental results

In this first part of the report we present results for a small test subset of the data.

We tested the method on a small and densely connected subset of term-advertiser data. The data consisted of 3000 unique bidded terms, 2000 unique advertisers and $92,344$ bids (non-zero elements in matrix).

Figure (1) shows the second (Fiedler) eigenvector of 3000x2000 term-advertiser matrix with elements sorted in increasing order; terms vector - top image, advertiser vector - bottom image. Staircase shows three distinct groups of nodes corresponding to three major clusters. In hierarchical spectral partitioning, we split this data into two groups using zero line as a separator.

Figure (2) presents ordering of 3000x2000 term-advertiser matrix; bidded terms are on y axis, advertisers - x axis. Left to right, top to bottom: initial data, after one path of spectral methods; after 3 levels of binary recursion using spectral; 8 levels; final ordering.

Finally, Figure (3) demonstrates spectral ordering of large data matrix.

Flat clusters can be obtained from partition tree by "cutting" it on a certain tree level. Table (1) shows data from several leaf nodes from the 8th level binary tree created by recursive spectral partitioning. Bidded phrase are shown one per line, advertiser are not shown; numbers indicates number of terms and advertiser per cluster (leaf).

5. Conclusions

In this report we demonstrate application of spectral graph partitioning methods to clustering of bidded term - advertiser datasets. We used "normalized cuts" approach to create balanced partitions and provided a way to create both hierarchical and flat clusters based on Fiedler vector. The major advantage of spectral method over other clustering techniques is its speed and scalability, which is guaranteed by efficient sparse matrix representation of the data and iterative eigenvalue computations algorithms.

The second part of this report will contain more implementation details of recursive procedure, details of experiments with various multi-partitioning schemes and results for the Overture's US market dataset.

Bibliography

[1] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[2] D. Beeferman and A. Berger. Agglomerative clustering of search engine query log. In *KDD*, 2000.

[3] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[4] David Cheng, Ravi Kannan, Santosh Vempala, and Grant Wang. On a recursive spectral algorithm for clustering from pairwise similarities.

[5] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD Conference, August 26 - 29*, pages 268–274, 2001.

[6] Chris H. Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of ICDM 2001*, pages 107–114, 2001.

[7] Drineas, Frieze, Kannan, Vempala, and Vinay. Clustering in large graphs and matrices. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1999.

[8] Anderson E., Bai Z., Bischof C., Blackford S., Demmel J., Dongarra J., Du Croz J., Greenbaum A., Hammarling S., McKenney A., and Sorensen D. *LAPACK Users' Guide*, third edition edition, 1999.

[9] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *Ai Magazine*, 17:37–54, 1996.

[10] Chris Fraley and Adrian E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588, 1998.

[11] Francisco M. Gomes and Danny C. Sorensen. *Arpack++. An object-oriented version of ARPACK eigenvalue package*, 2000.

[12] Guattery and Miller. On the performance of spectral graph partitioning methods. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1995.

[13] Stephen Guattery and Gary L. Miller. On the quality of spectral separators. *SIAM Journal on Matrix Analysis and Applications*, 19(3):701–719, 1998.

[14] Stephen Guattery and Gary L. Miller. Graph embeddings and laplacian eigenvalues. *SIAM Journal on Matrix Analysis and Applications*, 21(3):703–723, 2000.

[15] Bruce Hendrickson and Robert Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing*, 16(2):452–469, 1995.

[16] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice Hall, 1988.

[17] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral.

[18] Hagen L. and Kahng. A. B. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design*, 11:1074–1085, 1992.

[19] Fiedler Miroslav. Algebraic connectivity of graphs. *Czechosloval Mathematical Journal*, 23(98):298–305, 1973.

[20] Bojan Mohar. *Graph Theory, Combinatorics and Applications*, volume 2 of *Graph Theory, Combinatorics and Applications*. John Wiley & Sons, Inc, 1991.

[21] Bojan Mohar. Some applications of laplace eigenvalues of graphs, 1997.

[22] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm, 2001.

[23] Christian Posse. Hierarchical model-based clustering for large datasets. *Journal of Computational and Graphical Statistics*, 10(3):464–??, 2001.

[24] Lehoucq R.B., Sorensen D.C., and Yang C. *ARPACK User's Guide: Solution of Large Scaale Eigenvale Problems with Implicitly Restarted Arnoldi Methods*, 1997.

[25] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[26] Deepak Verma and Marina Meila. A comparison of spectral clustering algorithms.

[27] H. Zha, X. He, C. Ding, M. Gu, and H. Simon. Bipartite graph partitioning and data clustering. In *Proceedings of ACM CIKM 2001, the Tenth International Conference on Information and Knowledge Management, pp. 25-32, November 5-10*, 2001.

Node: 12 - 6	Node: 16 - 5	Node: 55 - 41	Node: 29-23
free card consolidation credit debt elimination consolidation debt free debt relief consolidate debt debt free card credit debt debt reduction counseling debt counseling credit bad credit	florist online flower rose flower online shop flower online send flower sympathy flower online order anniversary flower flower shop flower line flower funeral bouquet flower birthday flower flower online flower order buy flower day flower mother	multi vitamin d vitamin supplement vitamin b vitamin discount vitamin herb b complex vitamin e vitamin c vitamin vitamin herb natural calcium beta carotene acid amino herbal supplement b12 vitamin green tea ginseng antioxidant pycnogenol	optimization site web ranking site web promotion services site web promotion tool web promotion site tool web engine search site submission engine registration search engine search submission engine placement search service engine internet search site submission web registration url promotion services web site submit web engine search site submit submission url add url submit url service site submission web submission engine register search engine professional search .....

Table 1: Several leaf nodes the 8th level binary tree created by recursive spectral partitioning. Bidded phrase are shown one per line, advertiser are not shown; numbers indicates number of terms and advertiser per cluster (leaf node)

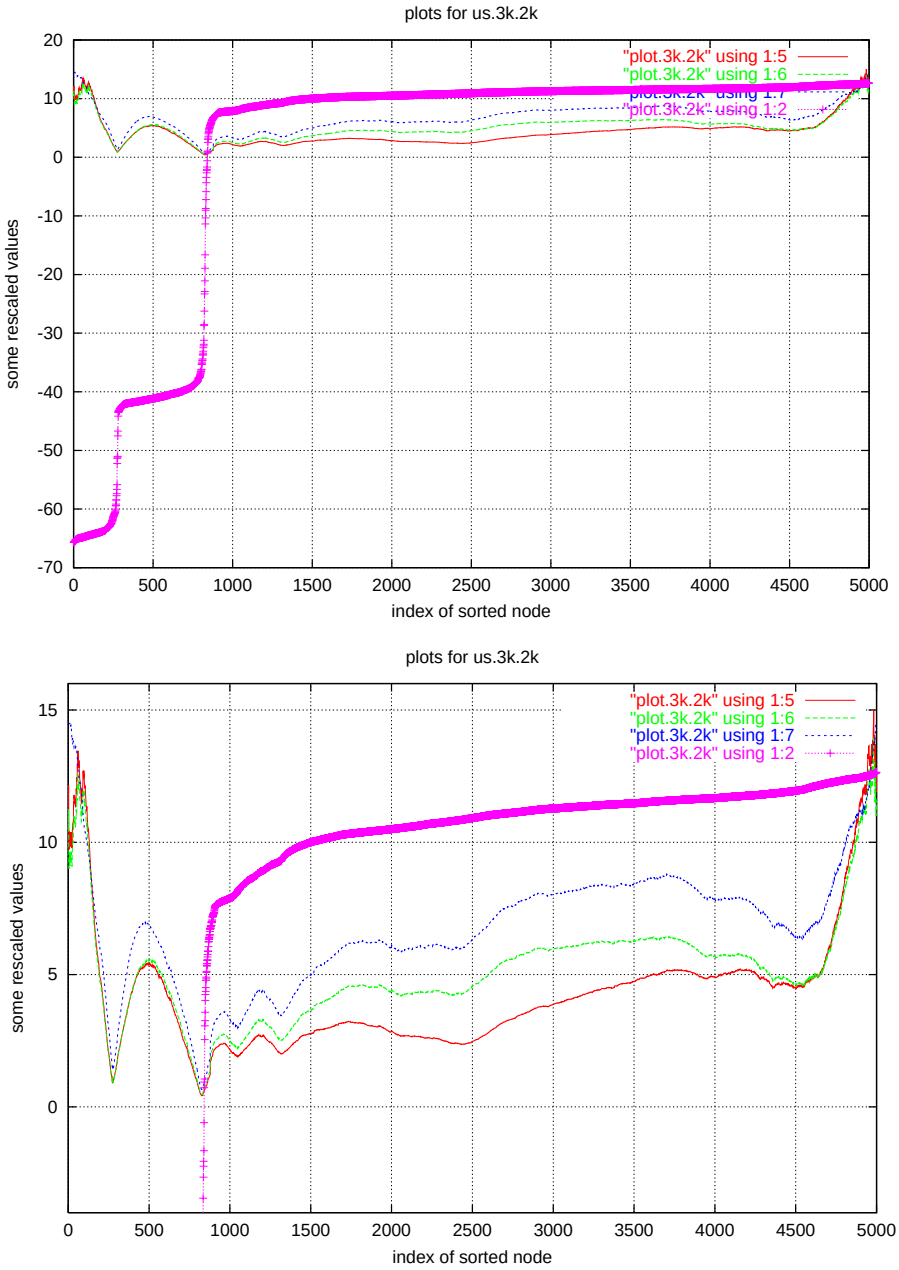

Figure 1: Second eigenvector of 3000x2000 term-advertiser matrix with elements sorted in increasing order plotted together with the values of several optimization functions. Terms and advertiser nodes are merged in one vector. Bottom image is zoomed in version of top. Note that abrupt changes in second eigenvector (gaps) correspond to the minima of partition function, thus to optimal cuts.

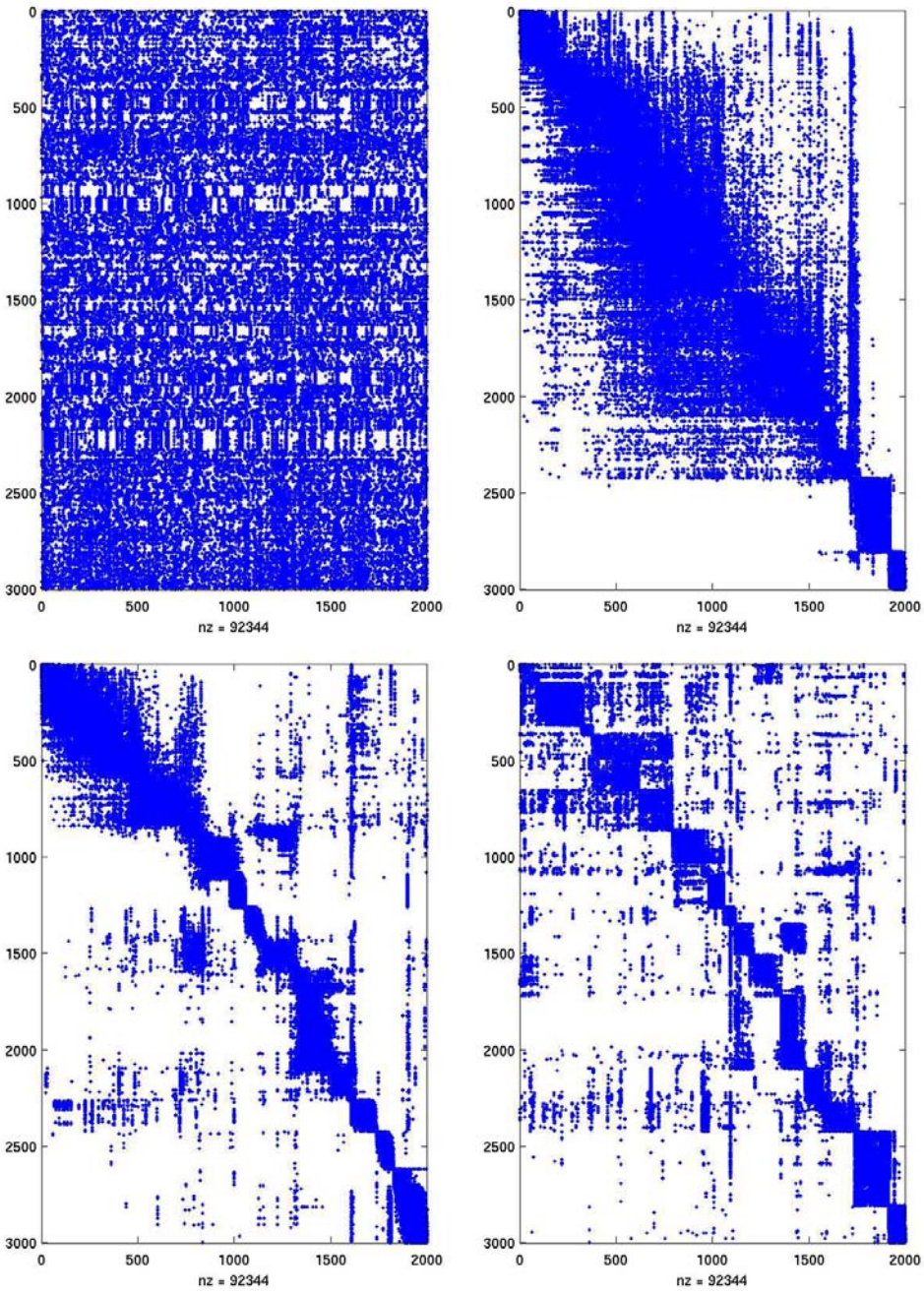

Figure 2: Ordering of 3000x2000 term-advertiser matrix; bidded terms are on y axis, advertisers - x axis. Left to right, top to bottom: initial data, after one path of spectral methods; after 3 levels of binary recursion using spectral; 8 levels; final ordering .

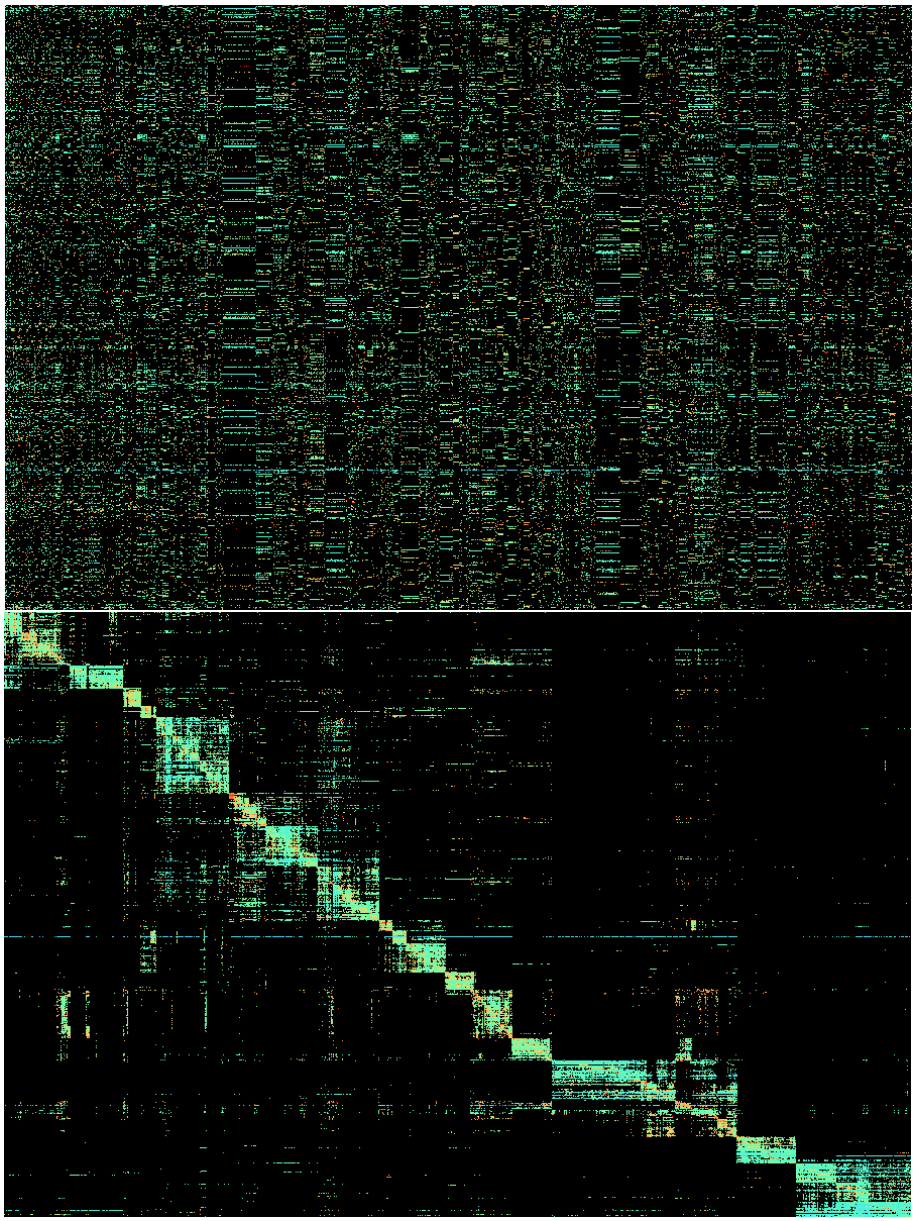

Figure 3: Spectral based ordering of term-advertiser matrix. Initial data matrix and matrix with rows and columns permuted according to spectral ordering