Co-clustering documents and words using Bipartite Spectral Graph Partitioning

Inderjit S. Dhillon
Department of Computer Sciences
University of Texas, Austin, TX 78712
inderjit@cs.utexas.edu

ABSTRACT

Both document clustering and word clustering are well studied problems. Most existing algorithms cluster documents and words separately but not simultaneously. In this paper we present the novel idea of modeling the document collection as a bipartite graph between documents and words, using which the simultaneous clustering problem can be posed as a bipartite graph partitioning problem. To solve the partitioning problem, we use a new spectral co-clustering algorithm that uses the second left and right singular vectors of an appropriately scaled word-document matrix to yield good bipartitionings. The spectral algorithm enjoys some optimality properties; it can be shown that the singular vectors solve a real relaxation to the NP-complete graph bipartitioning problem. We present experimental results to verify that the resulting co-clustering algorithm works well in practice.

1. INTRODUCTION

Clustering is the grouping together of similar objects. Given a collection of unlabeled documents, document clustering can help in organizing the collection thereby facilitating future navigation and search. A starting point for applying clustering algorithms to document collections is to create a *vector space model*[20]. The basic idea is (a) to extract unique content-bearing words from the set of documents treating these words as *features* and (b) to then represent each document as a vector in this feature space. Thus the entire document collection may be represented by a *word-by-document matrix* $\boldsymbol{A}$ whose rows correspond to words and columns to documents. A non-zero entry in $\boldsymbol{A}$, say A_{ij}, indicates the presence of word i in document j, while a zero entry indicates an absence. Typically, a large number of words exist in even a moderately sized set of documents, for example, in one test case we use 4303 words in 3893 documents. However, each document generally contains only a small number of words and hence, $\boldsymbol{A}$ is typically very sparse with almost 99% of the matrix entries being zero.

Existing document clustering methods include agglomerative clustering[25], the partitional k-means algorithm[7], projection based methods including LSA[21], self-organizing maps[18] and multidimensional scaling[16]. For computational efficiency required in on-line clustering, hybrid approaches have been considered such as in[5]. Graph-theoretic techniques have also been considered for clustering; many earlier hierarchical agglomerative clustering algorithms[9] and some recent work[3, 23] model the similarity between documents by a graph whose vertices correspond to documents and weighted edges or hyperedges give the similarity between vertices. However these methods are computationally prohibitive for large collections since the amount of work required just to form the graph is quadratic in the number of documents.

Words may be clustered on the basis of the documents in which they co-occur; such clustering has been used in the automatic construction of a statistical thesaurus and in the enhancement of queries[4]. The underlying assumption is that words that typically appear together should be associated with similar concepts. Word clustering has also been profitably used in the automatic classification of documents, see[1]. More on word clustering may be found in [24].

In this paper, we consider the problem of simultaneous or co-clustering of documents and words. Most of the existing work is on one-way clustering, i.e., either document or word clustering. A common theme among existing algorithms is to cluster documents based upon their word distributions while word clustering is determined by co-occurrence in documents. This points to a duality between document and term clustering. We pose this dual clustering problem in terms of finding minimum cut vertex partitions in a bipartite graph between documents and words. Finding a globally optimal solution to such a graph partitioning problem is NP-complete; however, we show that the second left and right singular vectors of a suitably normalized word-document matrix give an optimal solution to the real relaxation of this discrete optimization problem. Based upon this observation, we present a spectral algorithm that simultaneously partitions documents and words, and demonstrate that the algorithm gives good global solutions in practice.

A word about notation: small-bold letters such as $\boldsymbol{x}$, $\boldsymbol{u}$, $\boldsymbol{p}$ will denote column vectors, capital-bold letters such as $\boldsymbol{A}$, $\boldsymbol{M}$, $\boldsymbol{B}$ will denote matrices, and script letters such as $\mathcal{V}, \mathcal{D}, \mathcal{W}$ will usually denote vertex sets.

2. BIPARTITE GRAPH MODEL

First we introduce some relevant terminology about graphs. A graph $G = (\mathcal{V}, E)$ is a set of vertices $\mathcal{V} = \{1, 2, \ldots, |\mathcal{V}|\}$

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD 2001 San Francisco, California, USA
Copyright 2001 ACM X-XXXXX-XX-X/XX/XX ...$5.00.

and a set of edges $\{i,j\}$ each with edge weight E_{ij}. The adjacency matrix $\boldsymbol{M}$ of a graph is defined by

$$M_{ij} = \begin{cases} E_{ij}, & \text{if there is an edge } \{i,j\}, \\ 0, & \text{otherwise.} \end{cases}$$

Given a partitioning of the vertex set $\mathcal{V}$ into two subsets $\mathcal{V}_1$ and $\mathcal{V}_2$, the cut between them will play an important role in this paper. Formally,

$$\text{cut}(\mathcal{V}_1, \mathcal{V}_2) = \sum_{i \in \mathcal{V}_1, j \in \mathcal{V}_2} M_{ij}. \quad (1)$$

The definition of cut is easily extended to k vertex subsets,

$$\text{cut}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k) = \sum_{i<j} \text{cut}(\mathcal{V}_i, \mathcal{V}_j). \quad (2)$$

We now introduce our bipartite graph model for representing a document collection. An undirected bipartite graph is a triple $G = (\mathcal{D}, \mathcal{W}, E)$ where $\mathcal{D} = \{d_1, \ldots, d_n\}$, $\mathcal{W} = \{w_1, \ldots, w_m\}$ are two sets of vertices and E is the set of edges $\{\{d_i, w_j\} : d_i \in \mathcal{D}, w_j \in \mathcal{W}\}$. In our case $\mathcal{D}$ is the set of documents and $\mathcal{W}$ is the set of words they contain. An edge $\{d_i, w_j\}$ exists if word w_j occurs in document d_i; note that the edges are undirected. In this model, there are no edges between words or between documents.

An edge signifies an association between a document and a word. By putting positive weights on the edges, we can capture the strength of this association. One possibility is to have edge-weights equal term frequencies. In fact, most of the term-weighting formulae used in information retrieval may be used as edge-weights, see [20] for more details.

Consider the $m \times n$ word-by-document matrix $\boldsymbol{A}$ such that A_{ij} equals the edge-weight E_{ij}. It is easy to verify that the adjacency matrix of the bipartite graph may be written as

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{A} \\ \boldsymbol{A}^T & \boldsymbol{0} \end{bmatrix},$$

where we have ordered the vertices such that the first m vertices index the words while the last n index the documents.

We now show that the cut between different vertex subsets, as defined in (1) and (2), emerges naturally from our formulation of word and document clustering.

2.1 Simultaneous Clustering

A basic premise behind our algorithm is the observation:

Duality of word & document clustering: Word clustering induces document clustering while document clustering induces word clustering.

Given disjoint document clusters $\mathcal{D}_1, \ldots, \mathcal{D}_k$, the corresponding word clusters $\mathcal{W}_1, \ldots, \mathcal{W}_k$ may be determined as follows. A given word w_i belongs to the word cluster $\mathcal{W}_m$ if its association with the document cluster $\mathcal{D}_m$ is greater than its association with any other document cluster. Using our graph model, a natural measure of the association of a word with a document cluster is the sum of the edge-weights to all documents in the cluster. Thus,

$$\mathcal{W}_m = \left\{ w_i : \sum_{j \in \mathcal{D}_m} A_{ij} \geq \sum_{j \in \mathcal{D}_l} A_{ij}, \ \forall\, l = 1, \ldots, k \right\}.$$

Thus each of the word clusters is determined by the document clustering. Similarly given word clusters $\mathcal{W}_1, \ldots, \mathcal{W}_k$, the induced document clustering is given by

$$\mathcal{D}_m = \left\{ d_j : \sum_{i \in \mathcal{W}_m} A_{ij} \geq \sum_{i \in \mathcal{W}_l} A_{ij}, \ \forall\, l = 1, \ldots, k \right\}.$$

Note that this characterization is recursive in nature since document clusters determine word clusters, which in turn determine (better) document clusters. Clearly the "best" word and document clustering would correspond to a partitioning of the graph such that the crossing edges between partitions have minimum weight. This is achieved when

$$\text{cut}(\mathcal{W}_1 \cup \mathcal{D}_1, \ldots, \mathcal{W}_k \cup \mathcal{D}_k) = \min_{\mathcal{V}_1, \ldots, \mathcal{V}_k} \text{cut}(\mathcal{V}_1, \ldots, \mathcal{V}_k)$$

where $\mathcal{V}_1, \ldots, \mathcal{V}_k$ is any k-partitioning of the bipartite graph.

3. GRAPH PARTITIONING

Given a graph $G = (\mathcal{V}, E)$, the classical graph bipartitioning problem is to find nearly equally-sized vertex subsets $\mathcal{V}_1^*, \mathcal{V}_2^*$ of $\mathcal{V}$ such that $\text{cut}(\mathcal{V}_1^*, \mathcal{V}_2^*) = \min_{\mathcal{V}_1, \mathcal{V}_2} \text{cut}(\mathcal{V}_1, \mathcal{V}_2)$. Graph partitioning is an important problem and arises in various applications, such as circuit partitioning, telephone network design, load balancing in parallel computation, etc. However it is well known that this problem is NP-complete[12]. But many effective heuristic methods exist, such as, the Kernighan-Lin(KL)[17] and the Fiduccia-Mattheyses(FM)[10] algorithms. However, both the KL and FM algorithms search in the local vicinity of given initial partitionings and have a tendency to get stuck in local minima.

3.1 Spectral Graph Bipartitioning

Spectral graph partitioning is another effective heuristic that was introduced in the early 1970s[15, 8, 11], and popularized in 1990[19]. Spectral partitioning generally gives better global solutions than the KL or FM methods.

We now introduce the spectral partitioning heuristic. Suppose the graph $G = (\mathcal{V}, E)$ has n vertices and m edges. The $n \times m$ *incidence matrix* of G, denoted by $\boldsymbol{I_G}$ has one row per vertex and one column per edge. The column corresponding to edge $\{i,j\}$ of $\boldsymbol{I_G}$ is zero except for the i-th and j-th entries, which are $\sqrt{E_{ij}}$ and $-\sqrt{E_{ij}}$ respectively, where E_{ij} is the corresponding edge weight. Note that there is some ambiguity in this definition, since the positions of the positive and negative entries seem arbitrary. However this ambiguity will not be important to us.

DEFINITION 1. *The Laplacian matrix $\boldsymbol{L} = \boldsymbol{L_G}$ of G is an $n \times n$ symmetric matrix, with one row and column for each vertex, such that*

$$L_{ij} = \begin{cases} \sum_k E_{ik}, & i = j \\ -E_{ij}, & i \neq j \text{ and there is an edge } \{i,j\} \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

THEOREM 1. *The Laplacian matrix $\boldsymbol{L} = \boldsymbol{L_G}$ of the graph G has the following properties.*

1. *$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{M}$, where $\boldsymbol{M}$ is the adjacency matrix and $\boldsymbol{D}$ is the diagonal "degree" matrix with $D_{ii} = \sum_k E_{ik}$.*
2. *$\boldsymbol{L} = \boldsymbol{I_G} \boldsymbol{I_G}^T$.*
3. *$\boldsymbol{L}$ is a symmetric positive semi-definite matrix. Thus all eigenvalues of $\boldsymbol{L}$ are real and non-negative, and $\boldsymbol{L}$ has a full set of n real and orthogonal eigenvectors.*
4. *Let $\boldsymbol{e} = [1, \ldots, 1]^T$. Then $\boldsymbol{L}\boldsymbol{e} = \boldsymbol{0}$. Thus 0 is an eigenvalue of $\boldsymbol{L}$ and $\boldsymbol{e}$ is the corresponding eigenvector.*

5. *If the graph G has c connected components then $\boldsymbol{L}$ has c eigenvalues that equal 0.*
6. *For any vector $\boldsymbol{x}$, $\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} = \sum_{\{i,j\}\in E} E_{ij}(x_i - x_j)^2$.*
7. *For any vector $\boldsymbol{x}$, and scalars α and β*

$$(\alpha\boldsymbol{x} + \beta\boldsymbol{e})^T\boldsymbol{L}(\alpha\boldsymbol{x} + \beta\boldsymbol{e}) = \alpha^2\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x}. \quad (4)$$

Proof.

1. Part 1 follows from the definition of $\boldsymbol{L}$.
2. This is easily seen by multiplying $\boldsymbol{I_G}$ and $\boldsymbol{I_G}^T$.
3. By part 2, $\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{I_G}\boldsymbol{I_G}^T\boldsymbol{x} = \boldsymbol{y}^T\boldsymbol{y} \geq 0$, for all $\boldsymbol{x}$. This implies that $\boldsymbol{L}$ is symmetric positive semi-definite. All such matrices have non-negative real eigenvalues and a full set of n orthogonal eigenvectors[13].
4. Given any vector $\boldsymbol{x}$, $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{I_G}(\boldsymbol{I_G}^T\boldsymbol{x})$. Let k be the row of $\boldsymbol{I_G}^T\boldsymbol{x}$ that corresponds to the edge $\{i,j\}$, then it is easy to see that
$$(\boldsymbol{I_G}^T\boldsymbol{x})_k = \sqrt{E_{ij}}(x_i - x_j), \quad (5)$$
and so when $\boldsymbol{x} = \boldsymbol{e}$, $\boldsymbol{L}\boldsymbol{e} = \boldsymbol{0}$.
5. See [11].
6. This follows from equation (5).
7. This follows from part 4 above. □

For the rest of the paper, we will assume that the graph G consists of exactly one connected component. We now see how the eigenvalues and eigenvectors of $\boldsymbol{L}$ give us information about partitioning the graph. Given a bipartitioning of $\mathcal{V}$ into $\mathcal{V}_1$ and $\mathcal{V}_2$ ($\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$), let us define the partition vector $\boldsymbol{p}$ that captures this division,

$$p_i = \begin{cases} +1, & i \in \mathcal{V}_1, \\ -1, & i \in \mathcal{V}_2. \end{cases} \quad (6)$$

THEOREM 2. *Given the Laplacian matrix $\boldsymbol{L}$ of G and a partition vector $\boldsymbol{p}$, the Rayleigh Quotient*

$$\frac{\boldsymbol{p}^T\boldsymbol{L}\boldsymbol{p}}{\boldsymbol{p}^T\boldsymbol{p}} = \frac{1}{n} \cdot 4\ cut(\mathcal{V}_1, \mathcal{V}_2).$$

Proof. Clearly $\boldsymbol{p}^T\boldsymbol{p} = n$. By part 6 of Theorem 1, $\boldsymbol{p}^T\boldsymbol{L}\boldsymbol{p} = \sum_{\{i,j\}\in E} E_{ij}(p_i - p_j)^2$. Thus edges within $\mathcal{V}_1$ or $\mathcal{V}_2$ do not contribute to the above sum, while each edge between $\mathcal{V}_1$ and $\mathcal{V}_2$ contributes a value of 4 times the edge-weight. □

3.2 Eigenvectors as optimal partition vectors

Clearly, by Theorem 2, the cut is minimized by the trivial solution when all p_i are either -1 or +1. Informally, the cut captures the association between different partitions. We need an objective function that in addition to small cut values also captures the need for more "balanced" clusters.

We now present such an objective function. Let each vertex i be associated with a positive weight, denoted by weight(i), and let $\boldsymbol{W}$ be the diagonal matrix of such weights. For a subset of vertices $\mathcal{V}_l$ define its weight to be weight($\mathcal{V}_l$) = $\sum_{i\in\mathcal{V}_l}$ weight(i) = $\sum_{i\in\mathcal{V}_l} W_{ii}$. We consider subsets $\mathcal{V}_1$ and $\mathcal{V}_2$ to be "balanced" if their respective weights are equal. The following objective function favors balanced clusters,

$$\mathcal{Q}(\mathcal{V}_1, \mathcal{V}_2) = \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{\text{weight}(\mathcal{V}_1)} + \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{\text{weight}(\mathcal{V}_2)}. \quad (7)$$

Given two different partitionings with the same cut value, the above objective function value is smaller for the more balanced partitioning. Thus minimizing $\mathcal{Q}(\mathcal{V}_1, \mathcal{V}_2)$ favors partitions that have a small cut value and are balanced.

We now show that the Rayleigh Quotient of the following generalized partition vector $\boldsymbol{q}$ equals the above objective function value.

LEMMA 1. *Given graph G, let $\boldsymbol{L}$ and $\boldsymbol{W}$ be its Laplacian and vertex weight matrices respectively. Let $\eta_1 = weight(\mathcal{V}_1)$ and $\eta_2 = weight(\mathcal{V}_2)$. Then the generalized partition vector $\boldsymbol{q}$ with elements*

$$q_i = \begin{cases} +\sqrt{\frac{\eta_2}{\eta_1}}, & i \in \mathcal{V}_1, \\ -\sqrt{\frac{\eta_1}{\eta_2}}, & i \in \mathcal{V}_2, \end{cases}$$

satisfies $\boldsymbol{q}^T\boldsymbol{W}\boldsymbol{e} = 0$, and $\boldsymbol{q}^T\boldsymbol{W}\boldsymbol{q} = weight(\mathcal{V})$.

Proof. Let $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{e}$, then $y_i = \text{weight}(i) = W_{ii}$. Thus

$$\boldsymbol{q}^T\boldsymbol{W}\boldsymbol{e} = \sqrt{\frac{\eta_2}{\eta_1}} \sum_{i\in\mathcal{V}_1} \text{weight}(i) - \sqrt{\frac{\eta_1}{\eta_2}} \sum_{i\in\mathcal{V}_2} \text{weight}(i) = 0.$$

Similarly $\boldsymbol{q}^T\boldsymbol{W}\boldsymbol{q} = \sum_{i=1}^n W_{ii}q_i^2 = \eta_1 + \eta_2 = \text{weight}(\mathcal{V})$. □

THEOREM 3. *Using the notation of Lemma 1,*

$$\frac{\boldsymbol{q}^T\boldsymbol{L}\boldsymbol{q}}{\boldsymbol{q}^T\boldsymbol{W}\boldsymbol{q}} = \frac{cut(\mathcal{V}_1, \mathcal{V}_2)}{weight(\mathcal{V}_1)} + \frac{cut(\mathcal{V}_1, \mathcal{V}_2)}{weight(\mathcal{V}_2)}.$$

Proof. It is easy to show that the generalized partition vector $\boldsymbol{q}$ may be written as

$$\boldsymbol{q} = \frac{\eta_1 + \eta_2}{2\sqrt{\eta_1\eta_2}}\,\boldsymbol{p} + \frac{\eta_2 - \eta_1}{2\sqrt{\eta_1\eta_2}}\,\boldsymbol{e},$$

where $\boldsymbol{p}$ is the partition vector of (6). Using part 7 of Theorem 1, we see that

$$\boldsymbol{q}^T\boldsymbol{L}\boldsymbol{q} = \frac{(\eta_1 + \eta_2)^2}{4\eta_1\eta_2}\boldsymbol{p}^T\boldsymbol{L}\boldsymbol{p}.$$

Substituting the values of $\boldsymbol{p}^T\boldsymbol{L}\boldsymbol{p}$ and $\boldsymbol{q}^T\boldsymbol{W}\boldsymbol{q}$, from Theorem 2 and Lemma 1 respectively, proves the result. □

Thus to find the global minimum of (7), we can restrict our attention to generalized partition vectors of the form in Lemma 1. Even though this problem is still NP-complete, the following theorem shows that it is possible to find a real relaxation to the optimal generalized partition vector.

THEOREM 4. *The problem*

$$\min_{\boldsymbol{q}\neq\boldsymbol{0}} \frac{\boldsymbol{q}^T\boldsymbol{L}\boldsymbol{q}}{\boldsymbol{q}^T\boldsymbol{W}\boldsymbol{q}}, \quad \textit{subject to} \quad \boldsymbol{q}^T\boldsymbol{W}\boldsymbol{e} = 0,$$

is solved when $\boldsymbol{q}$ is the eigenvector corresponding to the 2nd smallest eigenvalue λ_2 of the generalized eigenvalue problem,

$$\boldsymbol{L}\boldsymbol{z} = \lambda\boldsymbol{W}\boldsymbol{z}. \quad (8)$$

Proof. This is a standard result from linear algebra[13]. □

3.3 Ratio-cut and Normalized-cut objectives

Thus far we have not specified the particular choice of vertex weights. A simple choice is to have weight(i) = 1 for all vertices i. This leads to the ratio-cut objective which has been considered in [14] (for circuit partitioning),

$$\text{Ratio-cut}(\mathcal{V}_1, \mathcal{V}_2) = \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{|\mathcal{V}_1|} + \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{|\mathcal{V}_2|}.$$

An interesting choice is to make the weight of each vertex equal to the sum of the weights of edges incident on it, i.e., weight(i) = $\sum_k E_{ik}$. This leads to the normalized-cut criterion that was used in [22] for image segmentation. Note that for this choice of vertex weights, the vertex weight matrix $\boldsymbol{W}$ equals the degree matrix $\boldsymbol{D}$, and weight($\mathcal{V}_i$) =

$\text{cut}(\mathcal{V}_1, \mathcal{V}_2) + \text{within}(\mathcal{V}_i)$ for $i = 1, 2$, where $\text{within}(\mathcal{V}_i)$ is the sum of the weights of edges with both end-points in $\mathcal{V}_i$. Then the normalized-cut objective function may be expressed as

$$\begin{aligned} \mathcal{N}(\mathcal{V}_1, \mathcal{V}_2) &= \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{\sum_{i \in \mathcal{V}_1} \sum_k E_{ik}} + \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{\sum_{i \in \mathcal{V}_2} \sum_k E_{ik}}, \\ &= 2 - \mathcal{S}(\mathcal{V}_1, \mathcal{V}_2), \\ \text{where} \quad \mathcal{S}(\mathcal{V}_1, \mathcal{V}_2) &= \frac{\text{within}(\mathcal{V}_1)}{\text{weight}(\mathcal{V}_1)} + \frac{\text{within}(\mathcal{V}_2)}{\text{weight}(\mathcal{V}_2)}. \end{aligned}$$

Note that $\mathcal{S}(\mathcal{V}_1, \mathcal{V}_2)$ measures the strengths of associations *within* each partition. Thus minimizing the normalized-cut is equivalent to maximizing the proportion of edge weights that lie within each partition.

4. THE SVD CONNECTION

In the previous section, we saw that the second eigenvector of the generalized eigenvalue problem $\boldsymbol{Lz} = \lambda \boldsymbol{Dz}$ provides a real relaxation to the discrete optimization problem of finding the minimum normalized cut. In this section, we present algorithms to find document and word clusterings using our bipartite graph model. In the bipartite case,

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{D_1} & -\boldsymbol{A} \\ -\boldsymbol{A}^T & \boldsymbol{D_2} \end{bmatrix}, \text{ and } \boldsymbol{D} = \begin{bmatrix} \boldsymbol{D_1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D_2} \end{bmatrix}$$

where $\boldsymbol{D_1}$ and $\boldsymbol{D_2}$ are diagonal matrices such that $D_1(i,i) = \sum_j A_{ij}$, $D_2(j,j) = \sum_i A_{ij}$. Thus $\boldsymbol{Lz} = \lambda \boldsymbol{Dz}$ may be written as

$$\begin{bmatrix} \boldsymbol{D_1} & -\boldsymbol{A} \\ -\boldsymbol{A}^T & \boldsymbol{D_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{D_1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \quad (9)$$

Assuming that both $\boldsymbol{D_1}$ and $\boldsymbol{D_2}$ are nonsingular, we can rewrite the above equations as

$$\begin{aligned} \boldsymbol{D_1}^{1/2}\boldsymbol{x} - \boldsymbol{D_1}^{-1/2}\boldsymbol{Ay} &= \lambda \boldsymbol{D_1}^{1/2}\boldsymbol{x}, \\ -\boldsymbol{D_2}^{-1/2}\boldsymbol{A}^T\boldsymbol{x} + \boldsymbol{D_2}^{1/2}\boldsymbol{y} &= \lambda \boldsymbol{D_2}^{1/2}\boldsymbol{y}. \end{aligned}$$

Letting $\boldsymbol{u} = \boldsymbol{D_1}^{1/2}\boldsymbol{x}$ and $\boldsymbol{v} = \boldsymbol{D_2}^{1/2}\boldsymbol{y}$, and after a little algebraic manipulation, we get

$$\begin{aligned} \boldsymbol{D_1}^{-1/2}\boldsymbol{AD_2}^{-1/2}\boldsymbol{v} &= (1-\lambda)\boldsymbol{u}, \\ \boldsymbol{D_2}^{-1/2}\boldsymbol{A}^T\boldsymbol{D_1}^{-1/2}\boldsymbol{u} &= (1-\lambda)\boldsymbol{v}. \end{aligned}$$

These are precisely the equations that define the singular value decomposition (SVD) of the normalized matrix $\boldsymbol{A_n} = \boldsymbol{D_1}^{-1/2}\boldsymbol{AD_2}^{-1/2}$. In particular, $\boldsymbol{u}$ and $\boldsymbol{v}$ are the left and right singular vectors respectively, while $(1-\lambda)$ is the corresponding singular value. Thus instead of computing the eigenvector of the second (smallest) eigenvalue of (9), we can compute the left and right singular vectors corresponding to the second (largest) singular value of $\boldsymbol{A_n}$,

$$\boldsymbol{A_n}\boldsymbol{v}_2 = \sigma_2 \boldsymbol{u}_2, \qquad \boldsymbol{A_n}^T\boldsymbol{u}_2 = \sigma_2 \boldsymbol{v}_2, \quad (10)$$

where $\sigma_2 = 1 - \lambda_2$. Computationally, working on $\boldsymbol{A_n}$ is much better since $\boldsymbol{A_n}$ is of size $w \times d$ while the matrix $\boldsymbol{L}$ is of the larger size $(w+d) \times (w+d)$.

The right singular vector $\boldsymbol{v}_2$ will give us a bipartitioning of documents while the left singular vector $\boldsymbol{u}_2$ will give us a bipartitioning of the words. By examining the relations (10) it is clear that this solution agrees with our intuition that a partitioning of documents should induce a partitioning of words, while a partitioning of words should imply a partitioning of documents.

4.1 The Bipartitioning Algorithm

The singular vectors $\boldsymbol{u}_2$ and $\boldsymbol{v}_2$ of $\boldsymbol{A_n}$ give a real approximation to the discrete optimization problem of minimizing the normalized cut. Given $\boldsymbol{u}_2$ and $\boldsymbol{v}_2$ the key task is to extract the optimal partition from these vectors.

The optimal generalized partition vector of Lemma 1 is two-valued. Thus our strategy is to look for a bi-modal distribution in the values of $\boldsymbol{u}_2$ and $\boldsymbol{v}_2$. Let m_1 and m_2 denote the bi-modal values that we are looking for. From the previous section, the second eigenvector of $\boldsymbol{L}$ is given by

$$\boldsymbol{z}_2 = \begin{bmatrix} \boldsymbol{D_1}^{-1/2}\boldsymbol{u}_2 \\ \boldsymbol{D_2}^{-1/2}\boldsymbol{v}_2 \end{bmatrix}. \quad (11)$$

One way to approximate the optimal bipartitioning is by the assignment of $\boldsymbol{z}_2(i)$ to the bi-modal values m_j $(j = 1, 2)$ such that the following sum-of-squares criterion is minimized,

$$\sum_{j=1}^{2} \sum_{\boldsymbol{z}_2(i) \in m_j} (\boldsymbol{z}_2(i) - m_j)^2.$$

The above is exactly the objective function that the classical k-means algorithm tries to minimize[9]. Thus we use the following algorithm to co-cluster words and documents:

Algorithm Bipartition

1. Given $\boldsymbol{A}$, form $\boldsymbol{A_n} = \boldsymbol{D_1}^{-1/2}\boldsymbol{AD_2}^{-1/2}$.
2. Compute the second singular vectors of $\boldsymbol{A_n}$, $\boldsymbol{u}_2$ and $\boldsymbol{v}_2$ and form the vector $\boldsymbol{z}_2$ as in (11).
3. Run the k-means algorithm on the 1-dimensional data $\boldsymbol{z}_2$ to obtain the desired bipartitioning.

The surprising aspect of the above algorithm is that we run k-means simultaneously on the reduced representations of both words and documents to get the co-clustering.

4.2 The Multipartitioning Algorithm

We can adapt our bipartitioning algorithm for the more general problem of finding k word and document clusters. One possibility is to use Algorithm Bipartition in a recursive manner. However, we favor a more direct approach. Just as the second singular vectors contain bi-modal information, the $\ell = \lceil \log_2 k \rceil$ singular vectors $\boldsymbol{u}_2, \boldsymbol{u}_3, \ldots, \boldsymbol{u}_{\ell+1}$, and $\boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_{\ell+1}$ often contain k-modal information about the data set. Thus we can form the ℓ-dimensional data set

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{D_1}^{-1/2}\boldsymbol{U} \\ \boldsymbol{D_2}^{-1/2}\boldsymbol{V} \end{bmatrix}, \quad (12)$$

where $\boldsymbol{U} = [\boldsymbol{u}_2, \ldots, \boldsymbol{u}_{\ell+1}]$, and $\boldsymbol{V} = [\boldsymbol{v}_2, \ldots, \boldsymbol{v}_{\ell+1}]$. From this reduced-dimensional data set, we look for the best k-modal fit to the ℓ-dimensional points $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k$ by assigning each ℓ-dimensional row, $\boldsymbol{Z}(i)$, to $\boldsymbol{m}_j$ such that the sum-of-squares

$$\sum_{j=1}^{k} \sum_{\boldsymbol{z}_2(i) \in m_j} \|\boldsymbol{Z}(i) - \boldsymbol{m}_j\|^2$$

is minimized. This can again be done by the classical k-means algorithm. Thus we obtain the following algorithm.

Algorithm Multipartition(k)

1. Given $\boldsymbol{A}$, form $\boldsymbol{A_n} = \boldsymbol{D_1}^{-1/2}\boldsymbol{AD_2}^{-1/2}$.
2. Compute $\ell = \lceil \log_2 k \rceil$ singular vectors of $\boldsymbol{A_n}$, $\boldsymbol{u}_2, \ldots \boldsymbol{u}_{\ell+1}$ and $\boldsymbol{v}_2, \ldots \boldsymbol{v}_{\ell+1}$, and form the matrix $\boldsymbol{Z}$ as in (12).
3. Run the k-means algorithm on the ℓ-dimensional data $\boldsymbol{Z}$ to obtain the desired k-way multipartitioning.

Name	# Docs	# Words	# Nonzeros($\boldsymbol{A}$)
MedCran	2433	5042	117987
MedCran_All	2433	17162	224325
MedCisi	2493	5447	109119
MedCisi_All	2493	19194	213453
Classic3	3893	4303	176347
Classic3_30docs	30	1073	1585
Classic3_150docs	150	3652	7960
Yahoo_K5	2340	1458	237969
Yahoo_K1	2340	21839	349792

Table 1: Details of the data sets

5. EXPERIMENTAL RESULTS

For some of our experiments, we used the popular Medline (1033 medical abstracts), Cranfield (1400 aeronautical systems abstracts) and Cisi (1460 information retrieval abstracts) collections. These document sets can be downloaded from ftp://ftp.cs.cornell.edu/pub/smart. For testing Algorithm Bipartition, we created mixtures consisting of 2 of these 3 collections. For example, MedCran contains documents from the Medline and Cranfield collections. Typically, we removed stop words, and words occurring in $< 0.2\%$ and $> 15\%$ of the documents. However, our algorithm has an in-built scaling scheme and is robust in the presence of large number of noise words, so we also formed word-document matrices by including all words, even stop words.

For testing Algorithm Multipartition, we created the Classic3 data set by mixing together Medline, Cranfield and Cisi which gives a total of 3893 documents. To show that our algorithm works well on small data sets, we also created subsets of Classic3 with 30 and 150 documents respectively.

Our final data set is a collection of 2340 Reuters news articles downloaded from Yahoo in October 1997[2]. The articles are from 6 categories: 142 from Business, 1384 from Entertainment, 494 from Health, 114 from Politics, 141 from Sports and 60 news articles from Technology. In the preprocessing, HTML tags were removed and words were stemmed using Porter's algorithm. We used 2 matrices from this collection: Yahoo_K5 contains 1458 words while Yahoo_K1 includes all 21839 words obtained after removing stop words. Details on all our test collections are given in Table 1.

5.1 Bipartitioning Results

In this section, we present bipartitioning results on the MedCran and MedCisi collections. Since we know the "true" class label for each document, the confusion matrix captures the goodness of document clustering. In addition, the measures of *purity* and *entropy* are easily derived from the confusion matrix[6].

Table 2 summarizes the results of applying Algorithm Bipartition to the MedCran data set. The confusion matrix at the top of the table shows that the document cluster $\mathcal{D}_0$ consists entirely of the Medline collection, while 1400 of the 1407 documents in $\mathcal{D}_1$ are from Cranfield. The bottom of Table 2 displays the "top" 7 words in each of the word clusters $\mathcal{W}_0$ and $\mathcal{W}_1$. The top words are those whose internal edge weights are the greatest. By the co-clustering, the word cluster $\mathcal{W}_i$ is associated with document cluster $\mathcal{D}_i$. It should be observed that the top 7 words clearly convey the "concept" of the associated document cluster.

Similarly, Table 3 shows that good bipartitions are also obtained on the MedCisi data set. Algorithm Bipartition uses the global spectral heuristic of using singular vectors which

	Medline	Cranfield
$\mathcal{D}_0$:	1026	0
$\mathcal{D}_1$:	7	1400
$\mathcal{W}_0$:	patients cells blood children hormone cancer renal	
$\mathcal{W}_1$:	shock heat supersonic wing transfer buckling laminar	

Table 2: Bipartitioning results for MedCran

	Medline	Cisi
$\mathcal{D}_0$:	970	0
$\mathcal{D}_1$:	63	1460
$\mathcal{W}_0$:	cells patients blood hormone renal rats cancer	
$\mathcal{W}_1$:	libraries retrieval scientific research science system book	

Table 3: Bipartitioning results for MedCisi

makes it robust in the presence of "noise" words. To demonstrate this, we ran the algorithm on the data sets obtained without removing even the stop words. The confusion matrices of Table 4 show that the algorithm is able to recover the original classes despite the presence of stop words.

5.2 Multipartitioning Results

In this section, we show that Algorithm Multipartition gives us good results. Table 5 gives the confusion matrix for the document clusters and the top 7 words of the associated word clusters found in Classic3. Note that since $k = 3$ in this case, the algorithm uses $\ell = \lceil \log_2 k \rceil = 2$ singular vectors for co-clustering.

As mentioned earlier, the Yahoo_K1 and Yahoo_K5 data sets contain 6 classes of news articles. Entertainment is the dominant class containing 1384 documents while Technology contains only 60 articles. Hence the classes are of varied sizes. Table 6 gives the multipartitioning result obtained by using $\ell = \lceil \log_2 k \rceil = 3$ singular vectors. It is clearly difficult to recover the original classes. However, the presence of many zeroes in the confusion matrix is encouraging. Table 6 shows that clusters $\mathcal{D}_1$ and $\mathcal{D}_2$ consist mainly of the Entertainment class, while $\mathcal{D}_4$ and $\mathcal{D}_5$ are "purely" from Health and Sports respectively. The word clusters show the underlying concepts in the associated document clusters (recall that the words are stemmed in this example). Table 7 shows that similar document clustering is obtained when fewer words are used.

Finally, Algorithm Multipartition does well on small collections also. Table 8 shows that even when mixing small (and random) subsets of Medline, Cisi and Cranfield our algorithm is able to recover these classes. This is in stark contrast to the spherical k-means algorithm that gives poor results on small document collections[7].

6. CONCLUSIONS

In this paper, we have introduced the novel idea of modeling a document collection as a bipartite graph using which we proposed a spectral algorithm for co-clustering words and documents. This algorithm has some nice theoretical properties as it provides the optimal solution to a real relaxation of the NP-complete co-clustering objective. In addition, our

	Medline	Cranfield
$\mathcal{D}_0$:	1014	0
$\mathcal{D}_1$:	19	1400

	Medline	Cisi
$\mathcal{D}_0$:	925	0
$\mathcal{D}_1$:	108	1460

Table 4: Results for MedCran_All and MedCisi_All

	Med	Cisi	Cran
$\mathcal{D}_0$:	965	0	0
$\mathcal{D}_1$:	65	1458	10
$\mathcal{D}_2$:	3	2	1390

$\mathcal{W}_0$: patients cells blood hormone renal cancer rats
$\mathcal{W}_1$: library libraries retrieval scientific science book system
$\mathcal{W}_2$: boundary layer heat shock mach supersonic wing

Table 5: Multipartitioning results for Classic3

	Bus	Entertain	Health	Politics	Sports	Tech
$\mathcal{D}_0$:	120	82	0	52	0	57
$\mathcal{D}_1$:	0	833	0	1	100	0
$\mathcal{D}_2$:	0	259	0	0	0	0
$\mathcal{D}_3$:	22	215	102	61	1	3
$\mathcal{D}_4$:	0	0	392	0	0	0
$\mathcal{D}_5$:	0	0	0	0	40	0

$\mathcal{W}_0$: clinton campaign senat house court financ white
$\mathcal{W}_1$: septemb tv am week music set top
$\mathcal{W}_2$: film emmi star hollywood award comedi fienne
$\mathcal{W}_3$: world health new polit entertain tech sport
$\mathcal{W}_4$: surgeri injuri undergo hospit england accord recommend
$\mathcal{W}_5$: republ advanc wildcard match abdelatif ac adolph

Table 6: Multipartitioning results for Yahoo_K1

algorithm works well on real examples as illustrated by our experimental results.

7. REFERENCES

[1] L. D. Baker and A. McCallum. Distributional clustering of words for text classification. In *ACM SIGIR*, pages 96–103, 1998.

[2] D. Boley. Hierarchical taxonomies using divisive partitioning. Technical Report TR-98-012, University of Minnesota, 1998.

[3] D. Boley, M. Gini, R. Gross, E.-H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the World Wide Web using WebACE. *AI Review*, 1998.

[4] C. J. Crouch. A cluster-based approach to thesaurus construction. In *ACM SIGIR*, pages 309–320, 1988.

[5] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *ACM SIGIR*, 1992.

[6] I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In V. K. R. Grossman, C. Kamath and R. Namburu, editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, 2001.

[7] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, January 2001. Also appears as IBM Research Report RJ 10147, July 1999.

[8] W. E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17:420–425, 1973.

[9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2000. 2nd Edition.

[10] C. M. Fiduccia and R. M. Mattheyses. A linear time heuristic for improving network partitions. Technical Report 82CRD130, GE Corporate Research, 1982.

[11] M. Fiedler. Algebraic connectivity of graphs. *Czecheslovak Mathematical Journal*, 23:298–305, 1973.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Company, 1979.

[13] G. H. Golub and C. F. V. Loan. *Matrix computations*. Johns Hopkins University Press, 3rd edition, 1996.

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

[15] K. M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 11(3):219–229, 1970.

[16] R. V. Katter. Study of document representations: Multidimensional scaling of indexing terms. System Development Corporation, Santa Monica, CA, 1967.

[17] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 29(2):291–307, 1970.

[18] T. Kohonen. *Self-organizing Maps*. Springer, 1995.

[19] A. Pothen, H. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, July 1990.

[20] G. Salton and M. J. McGill. *Introduction to Modern Retrieval*. McGraw-Hill Book Company, 1983.

[21] H. Schütze and C. Silverstein. Projections for efficient document clustering. In *ACM SIGIR*, 1997.

[22] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[23] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *AAAI 2000 Workshop on AI for Web Search*, 2000.

[24] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, second edition, 1979.

[25] E. M. Voorhees. *The effectiveness and efficiency of agglomerative hierarchic clustering in document retrieval*. PhD thesis, Cornell University, 1986.

	Bus	Entertain	Health	Politics	Sports	Tech
$\mathcal{D}_0$:	120	113	0	1	0	59
$\mathcal{D}_1$:	0	1175	0	0	136	0
$\mathcal{D}_2$:	19	95	4	73	5	1
$\mathcal{D}_3$:	1	6	217	0	0	0
$\mathcal{D}_4$:	0	0	273	0	0	0
$\mathcal{D}_5$:	2	0	0	40	0	0

$\mathcal{W}_0$: compani stock financi pr busi wire quote
$\mathcal{W}_1$: film tv emmi comedi hollywood previou entertain
$\mathcal{W}_2$: presid washington bill court militari octob violat
$\mathcal{W}_3$: health help pm death famili rate lead
$\mathcal{W}_4$: surgeri injuri undergo hospit england recommend discov
$\mathcal{W}_5$: senat clinton campaign house white financ republicn

Table 7: Multipartitioning results for Yahoo_K5

	Med	Cisi	Cran
$\mathcal{D}_0$:	9	0	0
$\mathcal{D}_1$:	0	10	0
$\mathcal{D}_2$:	1	0	10

	Med	Cisi	Cran
$\mathcal{D}_0$:	49	0	0
$\mathcal{D}_1$:	0	50	0
$\mathcal{D}_2$:	1	0	50

Table 8: Results for Classic3_30docs and Classic3_150docs