Co-clustering documents and words using Bipartite Spectral Graph Partitioning

UT CS Technical Report # TR 2001-05

Inderjit S. Dhillon
inderjit@cs.utexas.edu
http://www.cs.utexas.edu/users/inderjit
Department of Computer Sciences, University of Texas,
Austin, TX 78712-1188, USA.

March 2, 2001

Abstract

Both document clustering and word clustering are important and well-studied problems. By using the vector space model, a document collection may be represented as a *word-document* matrix. In this paper, we present the novel idea of modeling the document collection as a bipartite graph between documents and words. Using this model, we pose the clustering problem as a graph partitioning problem and give a new spectral algorithm that *simultaneously* yields a clustering of documents and words. This co-clustering algorithm uses the second left and right singular vectors of an appropriately scaled word-document matrix to yield good bipartitionings. In fact, it can be shown that these singular vectors give a real relaxation to the optimal solution of the graph bipartitioning problem. We present several experimental results to verify that the resulting co-clustering algorithm works well in practice and is robust in the presence of noise.

1 Introduction

Clustering is the grouping together of similar objects[18]. Given a collection of unlabeled documents, document clustering can help in organizing the collection thereby facilitating future navigation and search. Document clustering is a widely studied problem and many algorithms have been proposed for this task.

A starting point for applying clustering algorithms to document collections is to create a *vector space model*, alternatively known as a *bag-of-words model* [32]. 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 of certain weighted word frequencies in this feature space. Thus the entire document collection may be treated as 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 of our test cases we use 4303 words in 3893 documents. However, each document typically contains only a small number of words and hence, $\boldsymbol{A}$ is typically very sparse with almost 99% of the matrix entries being zero.

Using such a basic representation where words are features, many document clustering algorithms have been proposed. The various algorithms differ in the mathematical models they use and in their efficiency and scalability. The size of the collection to be clustered is an important variable. Clustering a large portion of the entire World Wide Web (such as in www.yahoo.com) is a much different proposition than clustering the smaller document collection returned by a web search engine (www.northernlight.com).

Existing document clustering methods include agglomerative clustering[40, 39, 29], the partitional k-means algorithm[8], projection based methods including LSA[2, 33], self-organizing maps[25, 21] and multidimensional scaling[27, 22]. For computational efficiency required in on-line clustering, hybrid approaches have been considered in[7, 19]. Recently there has been a flurry of activity in document clustering[3, 8, 30, 42]. Graph-theoretic techniques have also been considered for clustering; many earlier hierarchical agglomerative clustering algorithms[10] and some recent work[4, 37] model the similarity between documents by a graph whose vertices correspond to documents and weighted edges or hyperedges give the similarity between the vertices. A major drawback of these methods is that the work required just to form the graph is quadratic in the number of documents; thus these methods are computationally prohibitive for large document collections.

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[6]. 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]. A detailed treatment of some word clustering techniques is given in [38].

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.

We now give a brief outline of the paper. Section 2 introduces the bipartite graph model that we use for representing a document collection, and poses the co-clustering problem as one of finding the minimum cut in the bipartite graph. Constraints on the sizes of the clusters make this a graph partitioning problem which is known to be NP-complete. In Section 3, we introduce the spectral graph partitioning heuristic and show that the second eigenvector of a generalized eigenvalue problem gives the optimal solution to a real relaxation of the graph bipartitioning objective. In Section 4, we show that for the bipartite case, the dual clustering can be obtained from the second left and right singular vectors of a suitably normalized word-document matrix. Section 4.2 gives an algorithm that uses additional singular vectors to directly yield multiple document and word clusters. In Section 5, we present detailed experimental results and demonstrate that our co-clustering algorithm gives good solutions in practice. We discuss related work in Section 6. Finally, in Section 7 we present our conclusions and discuss future work.

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 $\mathcal{V} = \{1, 2, \ldots, |\mathcal{V}|\}$ and a set of edges $\{i, j\}$ each with edge weight E_{ij}. The adjacency matrix $\boldsymbol{M}$ of a graph is defined as

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

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,

$$\mathrm{cut}(\mathcal{V}_1, \mathcal{V}_2) = \sum_{i \in \mathcal{V}_1, j \in \mathcal{V}_2} M_{ij}. \tag{2.1}$$

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

$$\mathrm{cut}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k) = \sum_{i<j} \mathrm{cut}(\mathcal{V}_i, \mathcal{V}_j). \tag{2.2}$$

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 $\mathcal{W} = \{w_1, w_2, \ldots, w_m\}$ and $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ are two sets of vertices and E is the set of edges $\{\{w_i, d_j\} : w_i \in \mathcal{W}, d_j \in \mathcal{D}\}$. In our case, $\mathcal{D}$ corresponds to the set of documents and $\mathcal{W}$ corresponds to the set of words contained in the documents. There is an edge $\{w_i, d_j\}$ if word w_i occurs in document d_j; note that the edges are undirected. In this bipartite 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 degree of this association. One possibility is to have edge-weights equal term frequencies, i.e., the number of times a word occurs in a document. In fact, most of the term-weighting formulae used in information retrieval may be used as edge-weights, see [31, 32, 26] for more details. One popular term-weighting scheme is to have the edge-weight E_{ij} associated with the edge $\{w_i, d_j\}$ be

$$E_{ij} = t_{ij} \times \log\left(\frac{|\mathcal{D}|}{|\mathcal{D}_i|}\right),$$

where t_{ij} is the number of times word w_i occurs in document d_j, $|\mathcal{D}| = n$ is the total number of documents and $|\mathcal{D}_i|$ is the number of documents that contain word w_i.

Consider the $m \times n$ word-by-document matrix $\boldsymbol{A}$ such that A_{ij} equals the edge-weight E_{ij}. Then it can easily be seen that the adjacency matrix of the bipartite graph may be written as

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

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 (2.1) and (2.2), emerges naturally from our formulation of word and document clustering.

2.1 Simultaneous Clustering

A basic premise behind our algorithm is the following observation.

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

Given the document clusters $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k$ ($\cup_i \mathcal{D}_i = \mathcal{D}$ and $\mathcal{D}_i \cap \mathcal{D}_j = \phi, i \neq j$), the corresponding word clusters $\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_k$ ($\cup_i \mathcal{W}_i = \mathcal{W}$ and $\mathcal{W}_i \cap \mathcal{W}_j = \phi, i \neq j$) 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,

$$\mathcal{W}_m = \left\{ w_i : \sum_{j \in \mathcal{D}_m} A_{ij} \geq \sum_{j \in \mathcal{D}_l} A_{ij}, \quad \text{for all} \quad l = 1, 2, \ldots, k \right\}.$$

Thus each of the word clusters is determined by the document clustering. Similarly given a word clustering $\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_k$, the induced document clustering is given by

$$\mathcal{D}_m = \left\{ d_j : \sum_{i \in \mathcal{W}_m} A_{ij} \geq \sum_{i \in \mathcal{W}_l} A_{ij}, \quad \text{for all} \quad l = 1, 2, \ldots, k \right\}.$$

Note that this characterization is recursive in nature since a given document clustering determines a word clustering, which in turn determines a (better) document clustering. 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

$$\text{cut}(\mathcal{W}_1 \cup \mathcal{D}_1, \mathcal{W}_2 \cup \mathcal{D}_2, \ldots, \mathcal{W}_k \cup \mathcal{D}_k) = \min_{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k} \text{cut}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k)$$

where $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k$ is any partitioning of the bipartite graph into k vertex subsets.

Note that the above paragraph also hints at an "alternating partitioning" algorithm. However, such a formulation is too naive and typically leads to unbalanced cluster sizes. Indeed, the minimum cut of zero is obtained when all the vertices belong to one partition or cluster. Thus we need a mechanism to constrain the size of each cluster. In the next section, we pose an objective function which favors clusters that are "balanced" in addition to being "well-separated".

3 Graph Partitioning

Given a graph $G = (\mathcal{V}, E)$, the classical graph bipartitioning or bisection problem is to find nearly equally-sized vertex subsets $\mathcal{V}_1^*, \mathcal{V}_2^*$ of $\mathcal{V}$ such that

$$\text{cut}(\mathcal{V}_1^*, \mathcal{V}_2^*) \;=\; \min_{\mathcal{V}_1, \mathcal{V}_2} \text{cut}(\mathcal{V}_1, \mathcal{V}_2).$$

The graph partitioning problem is very important and arises in various applications, such as circuit partitioning and VLSI layout, telephone network design, load balancing in parallel computation, reordering sparse matrices, etc. However it is well known that this problem is NP-complete[13]. But many heuristic methods exist that are able to find a local minimum to this problem. Most effective among the earlier heuristics are the Kernighan-Lin(KL)[23] and the Fiduccia-Mattheyses(FM)[11] algorithms. However, these algorithms employ heuristics that search in the local vicinity of given initial partitionings and hence, have a tendency to get stuck in local minima.

3.1 Spectral Graph Bipartitioning

Spectral graph partitioning is another effective heuristic that was introduced by Hall[16], Donath & Hofmann[9] and Fiedler in the early 1970s[12], and popularized in 1990 by Pothen, Simon and Liou[28]. The spectral partitioning heuristic uses the second eigenvector of an associated matrix for bipartitioning the graph and has been found to give good global solutions that are better than the solutions obtained by the KL and FM algorithms.

We now introduce the spectral partitioning heuristic. Suppose the graph $G = (\mathcal{V}, E)$ has n vertices and m edges. The *incidence matrix* of G, denoted by $\boldsymbol{I_G}$ is an $n \times m$ matrix that has one row for each vertex and one column for each 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*

$$L_{ij} = \begin{cases} \sum_k E_{ik}, & i = j \\ -E_{ij}, & i \neq j \text{ and there is an edge } \{i, j\} \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

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 $\boldsymbol{e}$ be the vector of all ones, i.e., $\boldsymbol{e} = [1, 1, \ldots, 1]^T$. Then $\boldsymbol{Le} = \boldsymbol{0}$. 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*

$$(\alpha\boldsymbol{x} + \beta\boldsymbol{e})^T\boldsymbol{L}(\alpha\boldsymbol{x} + \beta\boldsymbol{e}) = \alpha^2\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x}. \tag{3.4}$$

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. Since $\boldsymbol{L} = \boldsymbol{I_G}\boldsymbol{I_G}^T$,

$$\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{I_G}\boldsymbol{I}_{\boldsymbol{G}}^T\boldsymbol{x} = \boldsymbol{y}^T\boldsymbol{y} \geq 0, \quad \text{for all } \boldsymbol{x}.$$

   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[14].

4. Given any vector $\boldsymbol{x}$, $\boldsymbol{Lx} = \boldsymbol{I_G}(\boldsymbol{I_G}^T\boldsymbol{x})$. 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

$$(\boldsymbol{I_G}^T\boldsymbol{x})_k \;=\; \sqrt{E_{ij}}(x_i - x_j), \tag{3.5}$$

   and so when $\boldsymbol{x} = \boldsymbol{e}$, $\boldsymbol{Le} = \boldsymbol{0}$.

5. See [12].

6. This follows from equation (3.5).

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, i.e., the second smallest eigenvalue of the Laplacian is nonzero (see part 5 of the above theorem). We now see how the eigenvalues and eigenvectors of $\boldsymbol{L}$ give us information about partitioning of 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,

$$p_i = \begin{cases} +1, & i \in \mathcal{V}_1, \\ -1, & i \in \mathcal{V}_2. \end{cases} \tag{3.6}$$

The following theorem shows that the Rayleigh Quotient of $\boldsymbol{p}$ with $\boldsymbol{L}$ is proportional to the cut induced by the partition vector $\boldsymbol{p}$.

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

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

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 real-valued 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 strength of the association between different partitions. We need an objective function that in addition to the need for 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,

$$W_{ij} = \begin{cases} \mathrm{weight}(i), & i = j, \\ 0, & i \neq j. \end{cases} \tag{3.7}$$

For any subset of vertices, $\mathcal{V}_l$, define its weight to be

$$\mathrm{weight}(\mathcal{V}_l) = \sum_{i \in \mathcal{V}_l} \mathrm{weight}(i) = \sum_{i \in \mathcal{V}_l} W_{ii}.$$

We will consider two subsets $\mathcal{V}_1$ and $\mathcal{V}_2$ to be "balanced" if their respective weights are nearly equal. The following objective function favors balanced clusters,

$$\mathcal{Q}(\mathcal{V}_1, \mathcal{V}_2) = \frac{\mathrm{cut}(\mathcal{V}_1, \mathcal{V}_2)}{\mathrm{weight}(\mathcal{V}_1)} + \frac{\mathrm{cut}(\mathcal{V}_1, \mathcal{V}_2)}{\mathrm{weight}(\mathcal{V}_2)}. \tag{3.8}$$

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 are balanced in addition to having a small cut value.

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

Lemma 1 *Given a graph G, let $\boldsymbol{L}$ and $\boldsymbol{W}$ be its Laplacian and vertex weight matrices respectively. Let $\eta_1 = \text{weight}(\mathcal{V}_1)$ and $\eta_2 = \text{weight}(\mathcal{V}_2)$. 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

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

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

$$\begin{aligned} \boldsymbol{q}^T \boldsymbol{W} \boldsymbol{e} &= \boldsymbol{q}^T \boldsymbol{y} = \sum_i q_i y_i, \\ &= \sqrt{\frac{\eta_2}{\eta_1}} \sum_{i \in \mathcal{V}_1} \text{weight}(i) - \sqrt{\frac{\eta_1}{\eta_2}} \sum_{i \in \mathcal{V}_2} \text{weight}(i), \\ &= \sqrt{\frac{\eta_2}{\eta_1}} \eta_1 - \sqrt{\frac{\eta_1}{\eta_2}} \eta_2 = 0. \end{aligned}$$

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

Theorem 3 *Using the notation of Lemma 1,*

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

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 (3.6). 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 globally minimum solution of the objective function (3.8), we can restrict our attention to generalized partition vectors of the form in Lemma 1. Even though this problem is still NP-complete, it is possible to find a real relaxation to this discrete optimization problem.

Theorem 4 *The problem*

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

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

$$\boldsymbol{L}\boldsymbol{z} \;=\; \lambda \boldsymbol{W}\boldsymbol{z}. \tag{3.9}$$

Proof. This is a standard result from linear algebra[14]. Note that $\boldsymbol{e}$ is the first eigenvector (corresponding to $\lambda = 0$) of (3.9) and thus the condition $\boldsymbol{q}^T \boldsymbol{W}\boldsymbol{e} = 0$ constrains the search for the optimal $\boldsymbol{q}$ to be over all vectors that are $\boldsymbol{W}$-orthogonal to the first eigenvector. □

Theorems 3 and 4 imply that the second eigenvector of (3.9) provides a real approximation to the optimal generalized partition vector. Thus the following corollary follows.

Corollary 1 *The second smallest eigenvalue of (3.9),* λ_2*, gives a lower bound on the objective function value in (3.8), i.e.,*

$$\mathcal{Q}(\mathcal{V}_1, \mathcal{V}_2) \;=\; \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{\text{weight}(\mathcal{V}_1)} + \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{\text{weight}(\mathcal{V}_2)} \;\geq\; \lambda_2.$$

3.3 Ratio-cut and Normalized-cut objectives

Thus far we have not specified the particular choice of vertex weights in (3.7). 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 [5, 15] (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 that are incident on it, i.e.,

$$\text{weight}(i) \;=\; \sum_k E_{ik}.$$

This leads to the normalized-cut criterion that was used in [34] for image segmentation. Note that for this choice of vertex weights, the vertex weight matrix $\boldsymbol{W}$ equals the degree matrix $\boldsymbol{D}$, and

$$\text{weight}(\mathcal{V}_i) = \text{cut}(\mathcal{V}_1, \mathcal{V}_2) + \text{within}(\mathcal{V}_i), \quad i = 1, 2,$$

where within$(\mathcal{V}_i)$ is the sum of the weights of edges that have 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 Spectral Bipartite Graph Partitioning with the SVD

In the previous section, we saw that the second eigenvector of the generalized eigenvalue problem,

$$\boldsymbol{L}\boldsymbol{z} \;=\; \lambda \boldsymbol{D}\boldsymbol{z}, \tag{4.10}$$

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

$$\begin{aligned} D_1(i,i) &= \sum_j A_{ij} \quad \text{(sum of edge-weights incident on word } i), \\ D_2(j,j) &= \sum_i A_{ij} \quad \text{(sum of edge-weights incident on document } j). \end{aligned}$$

Thus (4.10) may be written as

$$\begin{bmatrix} \boldsymbol{D_1} & -\boldsymbol{A} \\ -\boldsymbol{A}^T & \boldsymbol{D_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \;=\; \lambda \begin{bmatrix} \boldsymbol{D_1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \tag{4.11}$$

$$\begin{aligned} \Rightarrow \qquad \boldsymbol{D_1}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{y} &= \lambda \boldsymbol{D_1}\boldsymbol{x}, \\ \text{and} \quad -\boldsymbol{A}^T\boldsymbol{x} + \boldsymbol{D_2}\boldsymbol{y} &= \lambda \boldsymbol{D_2}\boldsymbol{y}. \end{aligned}$$

We assume that each document contains at least one word and each word is contained in at least one document. Thus both $\boldsymbol{D_1}$ and $\boldsymbol{D_2}$ are nonsingular and we can rewrite the above equations as

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

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

$$\boldsymbol{D_1}^{-1/2}\boldsymbol{A}\boldsymbol{D_2}^{-1/2}\boldsymbol{v} = (1-\lambda)\boldsymbol{u}, \;\text{ and }\; \boldsymbol{D_2}^{-1/2}\boldsymbol{A}^T\boldsymbol{D_1}^{-1/2}\boldsymbol{u} = (1-\lambda)\boldsymbol{v}.$$

These are precisely the equations that define the singular value decomposition (SVD) of the normalized matrix $\boldsymbol{A_n} = \boldsymbol{D_1}^{-1/2}\boldsymbol{A}\boldsymbol{D_2}^{-1/2}$. 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 (4.11), we can compute the left and right singular vectors corresponding to the second (largest) singular value of $\boldsymbol{A_n}$,

$$\boldsymbol{A_n v}_2 = \sigma_2 \boldsymbol{u}_2, \qquad \boldsymbol{A_n}^T \boldsymbol{u}_2 = \sigma_2 \boldsymbol{v}_2, \tag{4.12}$$

where $\sigma_2 = 1 - \lambda_2$. Computationally, working on $\boldsymbol{A_n}$ is superior 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 (4.12) 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

$$\boldsymbol{z}_2 = \begin{bmatrix} \boldsymbol{D_1}^{-1/2} \boldsymbol{u}_2 \\ \boldsymbol{D_2}^{-1/2} \boldsymbol{v}_2 \end{bmatrix}. \tag{4.13}$$

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[10]. Thus we can use the following algorithm to co-cluster words and documents into two clusters:

Algorithm Bipartition

1. Given $\boldsymbol{A}$, form $\boldsymbol{A_n} = \boldsymbol{D_1}^{-1/2} \boldsymbol{A} \boldsymbol{D_2}^{-1/2}$.
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 (4.13).
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 dimensional representations of both words and documents to get the co-clustering. In Section 5.1 we see that this algorithm yields good results in practice.

4.2 The Multipartitioning Algorithm

We can adapt our bipartitioning algorithm for the more general problem of finding k word and document clusters. One possible way to solve this multipartitioning problem 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, we have observed that the collection of 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

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{D_1}^{-1/2}\boldsymbol{U} \\ \boldsymbol{D_2}^{-1/2}\boldsymbol{V} \end{bmatrix}, \tag{4.14}$$

where

$$\boldsymbol{U} = [\boldsymbol{u}_2, \boldsymbol{u}_3, \ldots, \boldsymbol{u}_{\ell+1}], \quad \text{and} \quad \boldsymbol{V} = [\boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_{\ell+1}], \quad \ell = \lceil \log_2 k \rceil.$$

From this reduced-dimensional data set, we can 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 $\boldsymbol{A_n} = \boldsymbol{D_1}^{-1/2}\boldsymbol{A}\boldsymbol{D_2}^{-1/2}$.
2. Compute $\ell = \lceil \log_2 k \rceil$ singular vectors of $\boldsymbol{A_n}$, $\boldsymbol{u}_2, \boldsymbol{u}_3, \ldots \boldsymbol{u}_{\ell+1}$ and $\boldsymbol{v}_2, \boldsymbol{v}_3, \ldots \boldsymbol{v}_{\ell+1}$ and form the matrix $\boldsymbol{Z}$ as in (4.14).
3. Run the k-means algorithm on the ℓ-dimensional data $\boldsymbol{Z}$ to obtain the desired k-way multipartitioning.

5 Experimental Results

For some of our experimental results, we have used the popular MEDLINE, CISI, and CRANFIELD document sets. MEDLINE consists of 1033 abstracts from medical journals, CISI consists of 1460 abstracts from information retrieval papers, while CRANFIELD consists of 1400 abstracts from aeronautical systems papers. These document collections can be downloaded from ftp://ftp.cs.cornell.edu/pub/smart.

For testing Algorithm Bipartition, we created data sets that are a mixture of two of these three collections. For example, we created MEDCRAN which contains all documents in the MEDLINE and CRANFIELD collections. In creating the word-document matrices for these data sets, we typically removed stop words, and words occurring in less than 0.2%

Name	Description
MEDCRAN	MEDLINE & CRANFIELD: words occurring in $< .2\%$ and $> 15\%$ of documents are removed
MEDCRAN_ALL	MEDLINE & CRANFIELD: all words (including stop words) are included
MEDCISI	MEDLINE & CISI: words occurring in $< .2\%$ and $> 15\%$ of documents are removed
MEDCISI_ALL	MEDLINE & CISI: all words (including stop words) are included
CRANCISI	CRANFIELD & CISI: words occurring in $< .2\%$ and $> 15\%$ of documents are removed
CRANCISI_ALL	CRANFIELD & CISI: all words (including stop words) are included
CLASSIC3	MEDLINE, CRANFIELD & CISI: words occurring in $< .2\%$ and $> 15\%$ of documents are removed
CLASSIC3_30DOCS	MEDLINE, CRANFIELD & CISI: words occurring in $< .2\%$ and $> 15\%$ of documents are removed
CLASSIC3_150DOCS	MEDLINE, CRANFIELD & CISI: words occurring in $< .2\%$ and $> 15\%$ of documents are removed
CLASSIC3_300DOCS	MEDLINE, CRANFIELD & CISI: words occurring in $< .2\%$ and $> 15\%$ of documents are removed
YAHOO_K5	Reuters News Articles from Yahoo: words are stemmed and heavily pruned
YAHOO_K1	Reuters News Articles from Yahoo: words are stemmed and only stop words are pruned

Table 1: Description of the data sets

and greater than 15% of the documents. Most words occur very infrequently and so, this pruning step drastically reduces the number of words. For example, in MEDCRAN our pruning strategy removes 12120 out of a total of 17162 words. To show that our algorithms are robust in the presence of large number of words and noise words, we also used word-document matrices obtained by including all words, even stop words. Details on all our test collections are given in Tables 1 and 2.

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, 150 and 300 documents respectively.

Our final data set was used in [3] and is a collection of 2340 Reuters news articles downloaded from Yahoo in October 1997. The news 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. Both data sets were preprocessed by eliminating stop words and HTML tags and words were stemmed using Porter's algorithm[3]. The data set YAHOO_K5 contains 1458 words while YAHOO_K1 includes all 21839 words obtained after removing stop words. See Tables 1 and 2 for more details.

Name	# Documents	# Words	# Nonzeros in $\boldsymbol{A}$
MEDCRAN	2433	5042	117987
MEDCRAN_ALL	2433	17162	224325
MEDCISI	2493	5447	109119
MEDCISI_ALL	2493	19194	213453
CRANCISI	2860	4292	132989
CRANCISI_ALL	2860	14977	257762
CLASSIC3	3893	4303	176347
CLASSIC3_ALL	3893	22628	347770
CLASSIC3_30DOCS	30	1073	1585
CLASSIC3_150DOCS	150	3652	7960
CLASSIC3_300DOCS	300	5577	16077
YAHOO_K5	2340	1458	237969
YAHOO_K1	2340	21839	349792

Table 2: Details of the word-document matrices

5.1 Bipartitioning Results

In this section, we present bipartitioning results on the MEDCRAN, MEDCISI and CISICRAN collections. Since we know the "true" class label for each document, the confusion matrix is one way to capture the goodness of document clustering. In addition, we use the measures of *purity* and *entropy*. Suppose we are given c classes (true class labels) while the clustering algorithm produces k clusters. Cluster $\mathcal{D}_j$'s *purity* can be defined as

$$P(\mathcal{D}_j) = \frac{1}{n_j} \max_i (n_j^{(i)}),$$

where $n_j = |\mathcal{D}_j|$ and $n_j^{(i)}$ is the number of documents in $\mathcal{D}_j$ that belong to class i, $i = 1, \ldots, c$. Each cluster may contain samples from different classes. Purity gives the ratio of the dominant class size in the cluster to the cluster size itself. High purity means that the cluster is a "pure" subset of the dominant class. Additionally, we also use *entropy* as a quality measure, which is defined as follows:

$$H(\mathcal{D}_j) = -\frac{1}{\log c} \sum_{i=1}^{c} \frac{n_j^{(i)}}{n_j} \log\left(\frac{n_j^{(i)}}{n_j}\right).$$

Entropy is a more comprehensive measure than purity. It considers the distribution of classes in a cluster. Note that we have normalized entropy to take values between 0 and 1. An entropy value near 0 means the cluster is comprised entirely of 1 class, while an entropy value near 1 is bad since it implies that the cluster contains a uniform mixture of classes.

Table 3 gives the results of applying Algorithm Bipartition to the MEDCRAN data set. The confusion matrix at the top left of this table shows that the the document cluster $\mathcal{D}_0$ consists entirely of the MEDLINE collection, while $\mathcal{D}_1$ can clearly be identified with CRANFIELD. As a result, the purities of these clusters are nearly 1 while the entropies are close

	MEDLINE	CRANFIELD	Purity	Entropy
$\mathcal{D}_0$:	1026	0	1	0
$\mathcal{D}_1$:	7	1400	.995	.045

$\mathcal{W}_0$: patients cells blood children hormone cancer renal rats cell growth
$\mathcal{W}_1$: shock heat supersonic wing transfer buckling laminar plate hypersonic jet

Table 3: Bipartitioning result for MEDCRAN

	MEDLINE	CISI	Purity	Entropy
$\mathcal{D}_0$:	970	0	1	0
$\mathcal{D}_1$:	63	1460	.959	.248

$\mathcal{W}_0$: cells patients blood hormone renal rats cancer cell lens dna
$\mathcal{W}_1$: libraries retrieval scientific research science systems book computer indexing journals

Table 4: Bipartitioning result for MEDCISI

to 0. The bottom of Table 3 displays the "top" 10 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. Note that since Algorithm Bipartition co-clusters the documents and words, $\mathcal{W}_i$ is the word cluster associated with the document cluster $\mathcal{D}_i$. It should be observed that the top 10 words clearly convey the "concept" of the associated document cluster.

Similarly, Tables 4 and 5 show that good bipartitions are obtained using our spectral algorithm on the MEDCISI and CISICRAN data sets. Algorithm BIPARTITION uses the global spectral heuristic of using singular vectors which makes it robust in the presence of "noise" words. To show this, we ran the algorithm on the data sets obtained without removing even the stop words. The confusion matrices of Table 6 show that the algorithm is able to recover the original classes despite the presence of stop words. Note that the MEDLINE clusters are 100% pure in all cases.

	CISI	CRAN	Purity	Entropy
$\mathcal{D}_0$:	1457	12	.992	.068
$\mathcal{D}_1$:	3	1388	.998	.022

$\mathcal{W}_0$: libraries retrieval scientific science book systems indexing research literature journals
$\mathcal{W}_1$: layer heat shock mach supersonic wing buckling plate laminar hypersonic

Table 5: Bipartitioning result for CISICRAN

	MEDLINE	CRANFIELD
$\mathcal{D}_0$:	1014	0
$\mathcal{D}_1$:	19	1400

	MEDLINE	CISI
$\mathcal{D}_0$:	925	0
$\mathcal{D}_1$:	108	1460

	CISI	CRAN
$\mathcal{D}_0$:	1453	2
$\mathcal{D}_1$:	7	1398

Table 6: Confusion Matrices for MEDCRAN_ALL, MEDCISI_ALL and CISICRAN_ALL

	MED	CISI	CRAN	Purity	Entropy
$\mathcal{D}_0$:	966	0	0	1	0
$\mathcal{D}_1$:	66	1458	15	.947	.211
$\mathcal{D}_2$:	1	2	1385	.998	.015

$\mathcal{W}_0$:	patients cells blood hormone renal cancer rats cell disease lens
$\mathcal{W}_1$:	library libraries retrieval scientific science book systems system research indexing
$\mathcal{W}_2$:	boundary layer heat shock mach supersonic wing pressure buckling laminar

Table 7: Multipartitioning result for CLASSIC3

5.2 Multipartitioning Results

In this section, we show that Algorithm Multipartition gives us good results when multiple word and clusters are required. Table 7 gives the confusion matrix, purities, entropies of the document clusters and the top 10 words of the associated word clusters. Note that since $k = 3$ in this case, the algorithm uses $\ell = \lceil \log_2 k \rceil = 2$ singular vectors for co-clustering.

The YAHOO_K1 and YAHOO_K5 data sets contain 6 classes of news articles: Business, Entertainment, Health, Politics, Sports and Technology. Entertainment is the dominant class containing 1384 documents while Technology contains only 60 articles. Hence the sizes of the various classes are rather varied. Table 8 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 8 shows that clusters $\mathcal{D}_1$ and $\mathcal{D}_2$ can be identified with 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 9 shows that similar document clustering is obtained when fewer words are used.

Finally, Algorithm Multipartition does well on small collections also. Table 10 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[8].

6 Related Work

As mentioned in Section 1 both word and document clustering are well-studied problems. Our work addresses the duality between document clustering and word clustering. To the best of our knowledge, this co-clustering problem has not been studied previously. For

	Bus	Entertain	Health	Politics	Sports	Tech	Purity	Entropy
$\mathcal{D}_0$:	120	82	0	52	0	57	.386	.741
$\mathcal{D}_1$:	0	833	0	1	100	0	.892	.194
$\mathcal{D}_2$:	0	259	0	0	0	0	1	0
$\mathcal{D}_3$:	22	215	102	61	1	3	.532	.657
$\mathcal{D}_4$:	0	0	392	0	0	0	1	0
$\mathcal{D}_5$:	0	0	0	0	40	0	1	0

$\mathcal{W}_0$:	clinton campaign senat house court financ white compani reform stock
$\mathcal{W}_1$:	septemb tv am week music set top fridai record debut
$\mathcal{W}_2$:	film emmi star hollywood award comedi fienne henderson semler keener
$\mathcal{W}_3$:	world health new polit entertain tech sport scoreboard index bize
$\mathcal{W}_4$:	surgeri injuri undergo hospit england accord recommend twice headach heart
$\mathcal{W}_5$:	republi advanc wildcard match

Table 8: Multipartitioning result for YAHOO_K1

	Bus	Entertain	Health	Politics	Sports	Tech	Purity	Entropy
$\mathcal{D}_0$:	120	113	0	1	0	59	.410	.600
$\mathcal{D}_1$:	0	1175	0	0	136	0	.896	.186
$\mathcal{D}_2$:	19	95	4	73	5	1	.482	.638
$\mathcal{D}_3$:	1	6	217	0	0	0	.969	.084
$\mathcal{D}_4$:	0	0	273	0	0	0	1	0
$\mathcal{D}_5$:	2	0	0	40	0	0	.953	.085

$\mathcal{W}_0$:	compani stock financi pr busi wire quote percent industri gain
$\mathcal{W}_1$:	film tv emmi comedi hollywood previou entertain adult albert debut
$\mathcal{W}_2$:	presid washington bill court militari octob violat trade appeal secretari
$\mathcal{W}_3$:	health help pm death famili rate lead look driver people
$\mathcal{W}_4$:	surgeri injuri undergo hospit england recommend discov accord heart caus
$\mathcal{W}_5$:	senat clinton campaign house white financ republicn vote committee

Table 9: 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

	MED	CISI	CRAN
$\mathcal{D}_0$:	94	0	0
$\mathcal{D}_1$:	2	100	0
$\mathcal{D}_2$:	4	0	100

Table 10: Confusion matrices for CLASSIC3_30DOCS, CLASSIC3_150DOCS and CLASSIC3_150DOCS

general data matrices, there is some earlier work in [17] but this appears to be limited to matrices with small dimensions.

Our particular method of using spectral bipartite partitioning for clustering words or documents also appears to be new. The normalized cut criterion has been previously used in [34] for image segmentation. Eigenvectors and singular vectors have been used for document clustering previously, for example, the LSA-based approaches[2, 33], and the recent PDDP algorithm[3]. However, the LSA methods project document vectors onto much higher dimensional subspaces (100-300) and hence are computationally prohibitive. The PDDP algorithm is intimately related to inertial partitioning[41, 35] and uses the first principal component (PCA) to cluster documents. We believe that our normalization scheme that naturally arises from the normalized cut criterion enables us to use just the second left and right singular vectors for bipartitioning. When we used a different normalization scheme, such as the ones used in LSA and PDDP, our results were not as good, for example, we obtained poorer confusion matrices than in Table 6. We point out that it may be possible to show the optimality of our spectral partitioning scheme — a variant of our technique has been theoretically shown to give partitions that are close to optimal for a special class of graphs, such as planar graphs[36].

Recently there has been some work in obtaining document clusters by using graph partitioning[37]. However, this work uses a similarity graph model, where the vertices correspond to documents and edges correspond to document similarities obtained from a similarity measure such as cosine or generalized Jaccard similarity. However, these methods can be prohibitively expensive since just forming the similarity graph requires work that is quadratic in the number of documents.

7 Future Work and Conclusions

In this paper, we have introduced the novel idea of modeling a document collection as a bipartite graph between words and documents. Using this model, we pose the problem of co-clustering words and documents as a vertex partitioning problem in the bipartite graph. To solve this graph partitioning problem, we have used the spectral partitioning heuristic. We have shown that the second left and right singular vectors of a suitably normalized word-document matrix give a real relaxation to the discrete optimization problem. Multiple document and word clusters may be obtained from additional singular vectors. Our experimental results indicate that our spectral algorithm gives globally good solutions that are robust in the presence of noise words.

The results presented in this paper are for a new algorithm. In future work, we will conduct more detailed experiments on larger document collections and compare the efficiency and effectiveness of our co-clustering with other document and word clustering algorithms. To improve our spectral partitioning heuristic we plan to add the Kernighan-Lin or the Fiduccia-Mattheyses "smoothing"[23, 11]. Another avenue to explore would be multivelel methods for bipartite graph partitioning that minimize the normalized-cut criterion. The current Metis software[20] has important drawbacks: (i) it returns almost equal sized clusters, (ii) it optimizes a different objective function and (iii) it seems to accept only integer edge-weights. For web documents, bipartite graphs based on links have been used to enhance web search results[24]. As future work, we plan to combine words and links in a uniform manner for clustering web documents.

Our main contribution in this paper is the bipartite graph model for text collections which we have successfully applied to co-clustering. In the future, we will apply this bipartite model to other text mining problems, such as, text classification, better query retrieval, etc. Some fascinating aspects of this bipartite graph are that the vertex degrees for the words follow the Zipf distribution[43], and subgraphs appear to have a self-similar or fractal nature[8]. As future work, we intend to study various properties of these graphs. In addition, we feel that our co-clustering algorithm will be useful in many other applications, such as, in co-clustering DNA microarray and gene expression data, partitioning rectangular matrices for load balancing, etc.

References

[1] L. Douglas Baker and Andrew McCallum. Distributional clustering of words for text classification. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR*, pages 96–103. ACM, August 1998.

[2] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

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

[4] 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.

[5] C. K. Cheng and Y.-C. A. Wei. An improved two-way partitioning algorithm with stable performance. *IEEE Transactions on Computer-Aided Design*, 10:1502–1511, December 1991.

[6] C. J. Crouch. A Cluster-Based Approach to Thesaurus Construction. In *Proceedings of the 11th International Conference on Research and Development in Information Retrieval, SIGIR*, pages 309–320, Grenoble, France, June 1988.

[7] 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.

[8] 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.

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

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

[11] C. M. Fiduccia and R. M. Mattheyses. A linear time heuristic for improving network partitions. Technical Report 82CRD130, General Electric Co, Corporate Research and Development Center, Schenectady, NY, May 1982.

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

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

[14] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.

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

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

[17] J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, March 1972.

[18] J. A. Hartigan. *Clustering Algorithms*. Wiley, 1975.

[19] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In *ACM SIGIR*, pages 76–84, 1996.

[20] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1999.

[21] Samuel Kaski, Timo Honkela, Krista Lagus, and Teuvo Kohonen. WEBSOM—self-organizing maps of document collections. *Neurocomputing*, 21:101–117, 1998.

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

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

[24] Jon Kleinberg. Authoritative Sources in a Hyperlinked Environment. In *Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998. Extended version in Journal of the ACM 46(1999). Also appears as IBM Research Report RJ 10076, May 1997.

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

[26] T. G. Kolda. *Limited-Memory Matrix Methods with Applications*. PhD thesis, The Applied Mathematics Program, University of Maryland, College Park, Mayland, 1997.

[27] J. B. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29:115–129, 1964.

[28] Alex Pothen, Horst Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, July 1990.

[29] E. Rasmussen. Clustering algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 419–442. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[30] M. Sahami, S. Yusufali, and M. Baldonado. SONIA: A service for organizing networked information autonomously. In *Digital Libraries 98: Proceedings of the Third ACM Conference on Digital Libraries, New York, NY*, pages 200–209. ACM, 1999.

[31] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 4(5):513–523, 1988.

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

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

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

[35] Andrew Sohn and Horst D. Simon. S-HARP: A scalable parallel dynamic partitioner for adaptive mesh-based computations. In *Proceedings of Supercomputing 98, Orlando, Florida*, 1998.

[36] Daniel A. Spielman and Shang-Hua Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *37th Annual Symposium on Foundations of Computer Science*, pages 96–105, Burlington, Vermont, october 1996.

[37] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proceedings of the AAAI2000 Workshop on Artificial Intelligence for Web Search*, pages 58–64, Austin, Texas, July 2000. AAAI/MIT Press.

[38] C. J. van Rijsbergen. *Information Retrieval.* Butterworths, London, second edition, 1979. also available at http://www.dcs.gla.ac.uk/Keith/Preface.html.

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

[40] P. Willet. Recent trends in hierarchic document clustering: a critical review. *Information Processing & Management*, 24(5):577–597, 1988.

[41] R. Williams. Unification of spectral and inertial bisection. Technical Report CCSF-48, Center for Advanced Computing Research, California Institute of Technology, Pasadena, California, 1994.

[42] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *ACM SIGIR*, 1998.

[43] G. K. Zipf. *Human Behavior and the Principle of Least Effort.* Addison Wesley, Reading, MA, 1949.