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 bipartitions. In fact, it can be shown that these singular vectors give a real relaxation to the optimal solution of the graph partitioning 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 $A$ whose rows correspond to words and columns to documents. A non-zero entry in $A$, 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, \( A \) 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 \textit{www.yahoo.com}) is a much different proposition than clustering the smaller document
collection returned by a web search engine (\textit{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 doc-
uments 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 $x$, $u$, $p$ will denote column vectors, capital-bold letters such as $A$, $M$, $B$ 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 of a graph $M$ is defined as

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

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,

$$\text{cut}(\mathcal{V}_1, \mathcal{V}_2) = \sum_{i \in \mathcal{V}_1, j \in \mathcal{V}_2} M_{ij}. \hspace{1cm} (2.1)$$

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

$$\text{cut}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k) = \sum_{i < j} \text{cut}(\mathcal{V}_i, \mathcal{V}_j). \hspace{1cm} (2.2)$$

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 $d_j$, $|D| = n$ is the total number of documents and $|D_i|$ is the number of documents that contain word $w_i$.

Consider the $m \times n$ word-by-document matrix $A$ 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

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

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 $D_1, D_2, \ldots, D_k$ ($\cup_i D_i = D$ and $D_i \cap D_j = \emptyset$, $i \neq j$), the corresponding word clusters $W_1, W_2, \ldots, W_k$ ($\cup_i W_i = W$ and $W_i \cap W_j = \emptyset$, $i \neq j$) may be determined as follows. A given word $w_i$ belongs to the word cluster $W_m$ if its association with the document cluster $D_m$ 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,

$$W_m = \left\{ w_i : \sum_{j \in D_m} A_{ij} \geq \sum_{j \in D_l} A_{ij}, \text{ for all } l = 1, 2, \ldots, k \right\}.$$ 

Thus each of the word clusters is determined by the document clustering. Similarly given a word clustering $W_1, W_2, \ldots, W_k$, the induced document clustering is given by

$$D_m = \left\{ d_j : \sum_{i \in W_m} A_{ij} \geq \sum_{i \in W_l} A_{ij}, \text{ for all } l = 1, 2, \ldots, k \right\}.$$ 

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

$$\text{cut}(W_1 \cup D_1, W_2 \cup D_2, \ldots, W_k \cup D_k) = \min_{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k} \text{cut}({\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k}),$$

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 \( I_G \) 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 \( I_G \) 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 \( L = L_G \) of \( G \) is an \( n \times n \) symmetric matrix, with one row and column for each vertex, such that

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

**Theorem 1** The Laplacian matrix \( L = L_G \) of the graph \( G \) has the following properties.
1. $L = D - M$, where $M$ is the adjacency matrix and $D$ is the diagonal “degree” matrix with $D_{ii} = \sum_k E_{ik}$.

2. $L = I_G I_G^T$.

3. $L$ is a symmetric positive semi-definite matrix. Thus all eigenvalues of $L$ are real and non-negative, and $L$ has a full set of $n$ real and orthogonal eigenvectors.

4. Let $e$ be the vector of all ones, i.e., $e = [1, 1, \ldots, 1]^T$. Then $Le = 0$. Thus $0$ is an eigenvalue of $L$ and $e$ is the corresponding eigenvector.

5. If the graph $G$ has $c$ connected components then $L$ has $c$ eigenvalues that equal $0$.

6. For any vector $x$,

$$x^T L x = \sum_{(i,j) \in E} E_{ij} (x_i - x_j)^2.$$

7. For any vector $x$, and scalars $\alpha$ and $\beta$

$$(\alpha x + \beta e)^T L(\alpha x + \beta e) = \alpha^2 x^T L x. \quad (3.4)$$

Proof.

1. Part 1 follows from the definition of $L$.

2. This is easily seen by multiplying $I_G$ and $I_G^T$.

3. Since $L = I_G I_G^T$,

$$x^T L x = x^T I_G I_G^T x = y^T y \geq 0, \text{ for all } x.$$

This implies that $L$ 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 $x$, $Lx = I_G (I_G^T x)$. Let $k$ be the row of $I_G^T x$ that corresponds to the edge $\{i, j\}$, then it is easy to see that

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

and so when $x = e$, $Le = 0$.

5. See [12].

6. This follows from equation (3.5).

7. This follows from part 4 above. \qed
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 $L$ 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 $p$ that captures this division,

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

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

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

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

**Proof.** Clearly $p^T p = n$. By part 6 of Theorem 1,

$$p^T L p = \sum_{\{i,j\} \in E} E_{ij}(p_i - p_j)^2.$$ 

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. \qed

### 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 $W$ be the diagonal matrix of such weights,

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

For any subset of vertices, $\mathcal{V}_1$, define its weight to be

$$\text{weight}(\mathcal{V}_1) = \sum_{i \in \mathcal{V}_1} \text{weight}(i) = \sum_{i \in \mathcal{V}_1} W_{ii}.$$ 

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,

$$Q(\mathcal{V}_1, \mathcal{V}_2) = \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{\text{weight}(\mathcal{V}_1)} + \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{\text{weight}(\mathcal{V}_2)} \quad (3.8)$$
Given two different partitionings with the same cut value, the above objective function value is smaller for the more balanced partitioning. Thus minimizing $Q(V_1, V_2)$ 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 $q$ equals the above objective function value.

**Lemma 1** Given a graph $G$, let $L$ and $W$ be its Laplacian and vertex weight matrices respectively. Let $\eta_1 = \text{weight}(V_1)$ and $\eta_2 = \text{weight}(V_2)$. Then the generalized partition vector $q$ with elements

$$q_i = \begin{cases} + \sqrt{\frac{\eta_2}{\eta_1}}, & i \in V_1, \\ - \sqrt{\frac{\eta_1}{\eta_2}}, & i \in V_2, \end{cases}$$

satisfies

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

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

$$q^T W e = q^T y = \sum_i q_i y_i,$$

$$= \sqrt{\frac{\eta_2}{\eta_1}} \sum_{i \in V_1} \text{weight}(i) - \sqrt{\frac{\eta_1}{\eta_2}} \sum_{i \in V_2} \text{weight}(i),$$

$$= \sqrt{\frac{\eta_2}{\eta_1}} \eta_1 - \sqrt{\frac{\eta_1}{\eta_2}} \eta_2 = 0.$$ Similarly we can show that $q^T W q = \sum_{i=1}^{\eta} W_{ii} q_i^2 = \eta_1 + \eta_2 = \text{weight}(V).$ \hfill \Box

**Theorem 3** Using the notation of Lemma 1,

$$\frac{q^T L q}{q^T W q} = \frac{\text{cut}(V_1, V_2)}{\text{weight}(V_1)} + \frac{\text{cut}(V_1, V_2)}{\text{weight}(V_2)}.$$ 

**Proof.** It is easy to show that the generalized partition vector $q$ may be written as

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

where $p$ is the partition vector of (3.6). Using part 7 of Theorem 1, we see that

$$q^T L q = \frac{(\eta_1 + \eta_2)^2}{4\eta_1 \eta_2} p^T L p.$$ Substituting the values of $p^T L p$ and $q^T W q$, from Theorem 2 and Lemma 1 respectively, proves the result. \hfill \Box

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

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

is solved when $q$ is the eigenvector corresponding to the second smallest eigenvalue $\lambda_2$ of the generalized eigenvalue problem,

$$L z = \lambda W z. \quad (3.9)$$

Proof. This is a standard result from linear algebra [14]. Note that $e$ is the first eigenvector (corresponding to $\lambda = 0$) of (3.9) and thus the condition $q^T W e = 0$ constrains the search for the optimal $q$ to be over all vectors that are $W$-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), $\lambda_2$, gives a lower bound on the objective function value in (3.8), i.e.,

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

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 $W$ equals the degree matrix $D$, 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

$$N(\mathcal{V}_1, \mathcal{V}_2) = \frac{\text{cut} (\mathcal{V}_1, \mathcal{V}_2)}{\sum_{i \in \mathcal{V}_1} \sum_k E_{ik}} + \frac{\text{cut} (\mathcal{V}_1, \mathcal{V}_2)}{\sum_{i \in \mathcal{V}_2} \sum_k E_{ik}},$$

$$= 2 - S(\mathcal{V}_1, \mathcal{V}_2),$$

where

$$S(\mathcal{V}_1, \mathcal{V}_2) = \frac{\text{within}(\mathcal{V}_1)}{\text{weight}(\mathcal{V}_1)} + \frac{\text{within}(\mathcal{V}_2)}{\text{weight}(\mathcal{V}_2)}.$$
Note that $S(V_1, V_2)$ 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,

$$Lz = \lambda Dz,$$  \hspace{1cm} (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,

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

where $D_1$ and $D_2$ are diagonal matrices such that

$$D_1(i, i) = \sum_j A_{ij} \quad \text{(sum of edge-weights incident on word i)},$$

$$D_2(j, j) = \sum_i A_{ij} \quad \text{(sum of edge-weights incident on document j)}.$$  

Thus (4.10) may be written as

$$\begin{bmatrix} D_1 & -A \\ -A^T & D_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$  \hspace{1cm} (4.11)

$$\Rightarrow \quad D_1 x - A y = \lambda D_1 x,$$

and $$- A^T x + D_2 y = \lambda D_2 y.$$  

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

$$D_1^{1/2} x - D_1^{-1/2} A y = \lambda D_1^{1/2} x,$$

$$-D_2^{-1/2} A^T x + D_2^{1/2} y = \lambda D_2^{1/2} y.$$  

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

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

These are precisely the equations that define the singular value decomposition (SVD) of the normalized matrix $A_n = D_1^{-1/2} A D_2^{-1/2}$. In particular, $u$ and $v$ 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 \(A_n\),

\[
A_n v_2 = \sigma_2 u_2, \quad A_n^T u_2 = \sigma_2 v_2,
\]

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

The right singular vector \(v_2\) will give us a bipartitioning of documents while the left singular vector \(u_2\) 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 \(u_2\) and \(v_2\) of \(A_n\) give a real approximation to the discrete optimization problem of minimizing the normalized cut. Given \(u_2\) and \(v_2\) 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 \(u_2\) and \(v_2\). Let \(m_1\) and \(m_2\) denote the bi-modal values that we are looking for. From the previous section, the second eigenvector of \(L\) is given by

\[
z_2 = \begin{bmatrix} D_1^{-1/2} u_2 \\ D_2^{-1/2} v_2 \end{bmatrix},
\]

One way to approximate the optimal bipartitioning is by the assignment of \(z_2(i)\) to the bi-modal values \(m_j\) \((j = 1, 2)\) such that the following sum-of-squares criterion is minimized,

\[
\sum_{j=1}^{2} \sum_{z_2(i) = m_j} (z_2(i) - m_j)^2.
\]

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:

<table>
<thead>
<tr>
<th>Algorithm Bipartition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Given (A), form (A_n = D_1^{-1/2} A D_2^{-1/2}).</td>
</tr>
<tr>
<td>2. Compute the second singular vectors of (A_n), (u_2) and (v_2) and form the vector (z_2) as in (4.13).</td>
</tr>
<tr>
<td>3. Run the (k)-means algorithm on the 1-dimensional data (z_2) to obtain the desired bipartitioning.</td>
</tr>
</tbody>
</table>

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 \( u_2, u_3, \ldots, u_{\ell+1} \), and \( v_2, v_3, \ldots, v_{\ell+1} \) often contain \( k \)-modal information about the data set. Thus we can form the \( \ell \)-dimensional data set

\[
Z = \begin{bmatrix} D_1^{-1/2}U \\ D_2^{-1/2}V \end{bmatrix},
\]

(4.14)

where

\[
U = [u_2, u_3, \ldots, u_{\ell+1}], \quad \text{and} \quad V = [v_2, v_3, \ldots, v_{\ell+1}], \quad \ell = \lceil \log_2 k \rceil.
\]

From this reduced-dimensional data set, we can look for the best \( k \)-modal fit to the \( \ell \)-dimensional points \( m_1, \ldots, m_k \) by assigning each \( \ell \)-dimensional row, \( Z(i) \), to \( m_j \) such that the sum-of-squares

\[
\sum_{j=1}^{k} \sum_{z_2[i] \in m_j} ||Z(i) - m_j||^2
\]

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

<table>
<thead>
<tr>
<th>Algorithm Multipartition ((k))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Given ( A ), form ( A_n = D_1^{-1/2} A D_2^{-1/2} ).</td>
</tr>
<tr>
<td>2. Compute ( \ell = \lceil \log_2 k \rceil ) singular vectors of ( A_n, u_2, u_3, \ldots, u_{\ell+1} ) and ( v_2, v_3, \ldots, v_{\ell+1} ) and form the matrix ( Z ) as in (4.14).</td>
</tr>
<tr>
<td>3. Run the ( k )-means algorithm on the ( \ell )-dimensional data ( Z ) to obtain the desired ( k )-way multipartitioning.</td>
</tr>
</tbody>
</table>

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%
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MedCran</td>
<td>Medline &amp; Cranfield: words occurring in &lt; .2% and &gt; 15% of documents are removed</td>
</tr>
<tr>
<td>MedCran_all</td>
<td>Medline &amp; Cranfield: all words (including stop words) are included</td>
</tr>
<tr>
<td>MedCisi</td>
<td>Medline &amp; Cisi: words occurring in &lt; .2% and &gt; 15% of documents are removed</td>
</tr>
<tr>
<td>MedCisi_all</td>
<td>Medline &amp; Cisi: all words (including stop words) are included</td>
</tr>
<tr>
<td>CranCisi_all</td>
<td>Cranfield &amp; Cisi: all words (including stop words) are included</td>
</tr>
<tr>
<td>Classic3</td>
<td>Medline, Cranfield &amp; Cisi: words occurring in &lt; .2% and &gt; 15% of documents are removed</td>
</tr>
<tr>
<td>Classic3_30docs</td>
<td>Medline, Cranfield &amp; Cisi: words occurring in &lt; .2% and &gt; 15% of documents are removed</td>
</tr>
<tr>
<td>Classic3_150docs</td>
<td>Medline, Cranfield &amp; Cisi: words occurring in &lt; .2% and &gt; 15% of documents are removed</td>
</tr>
<tr>
<td>Classic3_300docs</td>
<td>Medline, Cranfield &amp; Cisi: words occurring in &lt; .2% and &gt; 15% of documents are removed</td>
</tr>
<tr>
<td>Yahoo_K5</td>
<td>Reuters News Articles from Yahoo: words are stemmed and heavily pruned</td>
</tr>
<tr>
<td>Yahoo_K1</td>
<td>Reuters News Articles from Yahoo: words are stemmed and only stop words are pruned</td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>Name</th>
<th># Documents</th>
<th># Words</th>
<th># Nonzeros in $A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MedCran</td>
<td>2463</td>
<td>5042</td>
<td>117987</td>
</tr>
<tr>
<td>MedCran_ALL</td>
<td>2463</td>
<td>17162</td>
<td>224325</td>
</tr>
<tr>
<td>MedCisi</td>
<td>2493</td>
<td>5447</td>
<td>109119</td>
</tr>
<tr>
<td>MedCisi_ALL</td>
<td>2493</td>
<td>19194</td>
<td>213453</td>
</tr>
<tr>
<td>CRANCisi</td>
<td>2860</td>
<td>4292</td>
<td>132989</td>
</tr>
<tr>
<td>CRANCisi_ALL</td>
<td>2860</td>
<td>14977</td>
<td>257762</td>
</tr>
<tr>
<td>CLASSIC3</td>
<td>3893</td>
<td>4303</td>
<td>176347</td>
</tr>
<tr>
<td>CLASSIC3_ALL</td>
<td>3893</td>
<td>22628</td>
<td>347770</td>
</tr>
<tr>
<td>CLASSIC3_30docs</td>
<td>30</td>
<td>1073</td>
<td>1585</td>
</tr>
<tr>
<td>CLASSIC3_150docs</td>
<td>150</td>
<td>3652</td>
<td>7960</td>
</tr>
<tr>
<td>CLASSIC3_300docs</td>
<td>300</td>
<td>5577</td>
<td>16077</td>
</tr>
<tr>
<td>Yahoo_K5</td>
<td>2340</td>
<td>1458</td>
<td>237969</td>
</tr>
<tr>
<td>Yahoo_K1</td>
<td>2340</td>
<td>21839</td>
<td>349792</td>
</tr>
</tbody>
</table>

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 $D_j$’s purity can be defined as

$$P(D_j) = \frac{1}{n_j} \max_{i} (n^{(i)}_j),$$

where $n_j = |D_j|$ and $n^{(i)}_j$ is the number of documents in $D_j$ 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:

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

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 $D_0$ consists entirely of the Medline collection, while $D_1$ can clearly be identified with Cranfield. As a result, the purities of these clusters are nearly 1 while the entropies are close
<table>
<thead>
<tr>
<th>Medline</th>
<th>Cranfield</th>
<th>Purity</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_0$:</td>
<td>1026</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$D_1$:</td>
<td>7</td>
<td>1400</td>
<td>.995</td>
</tr>
</tbody>
</table>

$W_0$: patients cells blood children hormone cancer renal rats cell growth

$W_1$: shock heat supersonic wing transfer buckling laminar plate hypersonic jet

Table 3: Bipartitioning result for MedCran

<table>
<thead>
<tr>
<th>Medline</th>
<th>Cisi</th>
<th>Purity</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_0$:</td>
<td>970</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$D_1$:</td>
<td>63</td>
<td>1460</td>
<td>.959</td>
</tr>
</tbody>
</table>

$W_0$: cells patients blood hormone renal rats cancer cell lens dna

$W_1$: 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 $W_0$ and $W_1$. The top words are those whose internal edge weights are the greatest. Note that since Algorithm Bipartition co-clusters the documents and words, $W_i$ is the word cluster associated with the document cluster $D_i$. 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.

<table>
<thead>
<tr>
<th>Cisi</th>
<th>Cran</th>
<th>Purity</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_0$:</td>
<td>1457</td>
<td>12</td>
<td>.992</td>
</tr>
<tr>
<td>$D_1$:</td>
<td>3</td>
<td>1388</td>
<td>.998</td>
</tr>
</tbody>
</table>

$W_0$: libraries retrieval scientific science book systems indexing research literature journals

$W_1$: layer heat shock mach supersonic wing buckling plate laminar hypersonic

Table 5: Bipartitioning result for CisiCran
Table 6: Confusion Matrices for MedCran_All, MedCisi_All and CisiCran_All

<table>
<thead>
<tr>
<th></th>
<th>Med</th>
<th>Cisi</th>
<th>Cran</th>
<th>Purity</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathcal{D}_0)</td>
<td>966</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(\mathcal{D}_1)</td>
<td>66</td>
<td>1458</td>
<td>15</td>
<td>.947</td>
<td>.211</td>
</tr>
<tr>
<td>(\mathcal{D}_2)</td>
<td>1</td>
<td>2</td>
<td>1385</td>
<td>.998</td>
<td>.015</td>
</tr>
</tbody>
</table>

\(\mathcal{W}_0\): 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

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
<table>
<thead>
<tr>
<th></th>
<th>Bus</th>
<th>Entertain</th>
<th>Health</th>
<th>Politics</th>
<th>Sports</th>
<th>Tech</th>
<th>Purity</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_0$</td>
<td>120</td>
<td>82</td>
<td>0</td>
<td>52</td>
<td>0</td>
<td>57</td>
<td>.386</td>
<td>.741</td>
</tr>
<tr>
<td>$D_1$</td>
<td>0</td>
<td>833</td>
<td>0</td>
<td>1</td>
<td>100</td>
<td>0</td>
<td>.892</td>
<td>.194</td>
</tr>
<tr>
<td>$D_2$</td>
<td>0</td>
<td>259</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$D_3$</td>
<td>22</td>
<td>215</td>
<td>102</td>
<td>61</td>
<td>1</td>
<td>3</td>
<td>.532</td>
<td>.657</td>
</tr>
<tr>
<td>$D_4$</td>
<td>0</td>
<td>0</td>
<td>392</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$D_5$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$W_0$: clinton campaign senat house court financ white compani reform stock
$W_1$: septemb tv am week music set top fridai record debut
$W_2$: film emmi star hollywood award comedie fienne henderson semler keener
$W_3$: world health new polit entertain tech sport scoreboard index bize
$W_4$: surgeri injuri undergo hospit england accord recommend twice headach heart
$W_5$: republi advanc wildcard match

Table 8: Multipartitioning result for Yahoo_K1

<table>
<thead>
<tr>
<th></th>
<th>Bus</th>
<th>Entertain</th>
<th>Health</th>
<th>Politics</th>
<th>Sports</th>
<th>Tech</th>
<th>Purity</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_0$</td>
<td>120</td>
<td>113</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>59</td>
<td>.410</td>
<td>.600</td>
</tr>
<tr>
<td>$D_1$</td>
<td>0</td>
<td>1175</td>
<td>0</td>
<td>0</td>
<td>136</td>
<td>0</td>
<td>.896</td>
<td>.186</td>
</tr>
<tr>
<td>$D_2$</td>
<td>19</td>
<td>95</td>
<td>4</td>
<td>73</td>
<td>5</td>
<td>1</td>
<td>.482</td>
<td>.638</td>
</tr>
<tr>
<td>$D_3$</td>
<td>1</td>
<td>6</td>
<td>217</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.969</td>
<td>.084</td>
</tr>
<tr>
<td>$D_4$</td>
<td>0</td>
<td>0</td>
<td>273</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$D_5$</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>.953</td>
<td>.085</td>
</tr>
</tbody>
</table>

$W_0$: compani stock financ pr busi wire quote percent industri gain
$W_1$: film tv emmi comedie hollywood previou entertain adult albert debut
$W_2$: presid washington bill court militari octob violat trade appeal secretari
$W_3$: health help pm death famili rate lead look driver people
$W_4$: surgeri injuri undergo hospit england accord recommend discov accord heart caus
$W_5$: senat clinton campaign house white financ republicin vote committee

Table 9: Multipartitioning results for Yahoo_K5

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 multilevel 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


