A Divide-and-Merge Methodology for Clustering

DAVID CHENG
Massachusetts Institute of Technology
RAVI KANNAN
Yale University
SANTOSH VEMPALA
Massachusetts Institute of Technology
and
GRANT WANG
Massachusetts Institute of Technology

We present a divide-and-merge methodology for clustering a set of objects that combines a top-down "divide" phase with a bottom-up "merge" phase. In contrast, previous algorithms use either top-down or bottom-up methods to construct a hierarchical clustering or produce a flat clustering using local search (e.g., k-means). For the divide phase, which produces a tree whose leaves are the elements of the set, we suggest an efficient spectral algorithm. When the data is in the form of a sparse document-term matrix, we show how to modify the algorithm so that it maintains sparsity and runs in linear space. The merge phase quickly finds the optimal partition that respects the tree for many natural objective functions, e.g., k-means, min-diameter, min-sum, correlation clustering, etc.. We present a thorough experimental evaluation of the methodology. We describe the implementation of a meta-search engine that uses this methodology to cluster results from web searches. We also give comparative empirical results on several real datasets.

Categories and Subject Descriptors: H. Information Systems [**H.3 Information Retrieval**]: H.3.3 Information Search and Retreival

Additional Key Words and Phrases: Clustering, data mining, information retrieval

1. INTRODUCTION

The rapidly increasing volume of readily accessible data presents a challenge for computer scientists: find methods that can locate relevant information and organize it in an intelligible way. This is different from the classical database problem in at least two ways: first, there may neither be the time nor (in the long term) the computer memory to store and structure all the data in a central location. Second, one would like to find interesting patterns in the data without knowing what to look for in advance.

Clustering refers to the process of classifying a set of data objects into groups so that each group consists of similar objects and objects from different groups are dissimilar. The classification could either be flat (a partition of the data) or hierarchical [JD88]. Clustering has been proposed as a method to aid information retrieval in many contexts (e.g. [CKPT92; VR79; SKK00; LA99; Dhi01]). Document clustering can help generate a hierarchical taxonomy efficiently (e.g. [Bol98; ZK02]) as well as organize the results of a web search (e.g. [ZEMK97; WF00]). It has also been used to learn (or fit) mixture models to datasets [Hof99] and for image segmentation [TG97].

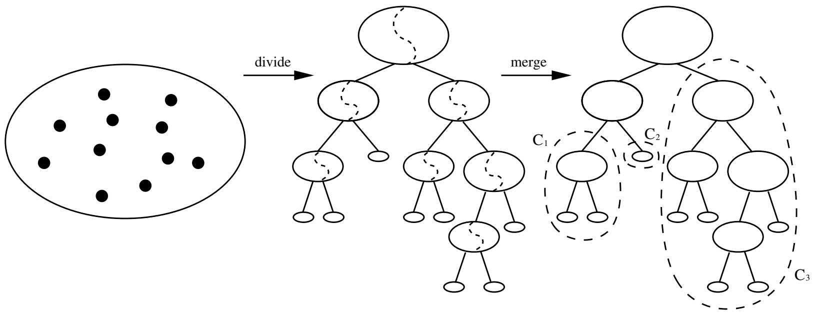


Fig. 1. The Divide-and-Merge methodology

Most hierarchical clustering algorithms can be described as either divisive methods (i.e. top-down) or agglomerative methods (i.e. bottom-up) [And73; JD88; JMF99]. Both methods create trees, but do not provide a flat clustering. A divisive algorithm begins with the entire set and recursively partitions it into two pieces, forming a tree. An agglomerative algorithm starts with each object in its own cluster and iteratively merges clusters. We combine top-down and bottom-up techniques to create both a hierarchy and a flat clustering. In the divide phase, we can apply any divisive algorithm to form a tree T whose leaves are the objects. This is followed by the merge phase in which we start with each leaf of T in its own cluster and merge clusters going up the tree. The final clusters form a partition of the dataset and are tree-respecting clusters, i.e., they are subtrees rooted at some node of T. For many natural objective functions, the merge phase can be executed optimally, producing the best tree-respecting clustering. Figure 1 shows a depiction of the methodology.

For the divide phase we suggest using the theoretical spectral algorithm studied in [KVV04]. The algorithm is well-suited for clustering objects with pairwise similarities – in [KVV04], the authors prove that the hierarchical tree formed by the divide phase contains a "good clustering". Unfortunately, the input to that algorithm is a matrix of pairwise similarities; for a dataset with n objects, the running time could be $O(n^4)$. We describe an efficient implementation of the spectral algorithm when the data is presented as a document-term matrix and the similarity function is the inner product. Note that the document-term matrix is often sparse, and thus is significantly smaller than the matrix of pairwise similarities; our implementation maintains the sparsity of the input. For a document-term matrix for n objects with M nonzeros, our implementation runs in $O(Mn \log n)$ in the worst case and seems to perform much better in practice (see Figure 3(a)). The algorithm uses space linear in the number of nonzeros M. The data need not be text; all that is needed is for the similarity of two objects to be the inner product between the two vectors representing the objects.

The class of functions for which the merge phase can find an optimal tree-respecting clustering include standard objectives such as k-means [HW79], min-diameter [CFM97], and min-sum [SG76], as well as correlation clustering, a formulation of clustering that has seen recent interest [BBC02; CGW03; DI03; EF03;

Swa04]. By optimal tree-respecting clustering, we mean that the clustering found by the merge phase is optimal over the set of clusterings that respect the tree, i.e. clusterings whose clusters are nodes in the tree. Note that optimizing each of the standard objective functions is NP-hard. Although approximation algorithms exist for these problems, many of them have impractical running times. Our methodology can be seen as an efficient alternative.

We conducted a thorough experimental evaluation for the methodology. The first evaluation is in the form of a meta-search engine, EigenCluster [CKVW], that clusters the results of a query to a standard web search engine. EigenCluster consistently finds the natural clustering for queries that exhibit polysemy, e.g., for the query `monte carlo`, EigenCluster finds clusters pertaining to the car model, the city in Monaco, and the simulation technique. We describe EigenCluster and show results of example queries in Section 3.

We apply the methodology to clustering real-world datasets: text, gene expression, and categorical data. In Section 4.2, we describe the results of a suite of experiments that test the effectiveness of the spectral algorithm as a procedure for the divide phase. For these datasets, the "true" clustering is known in advance. We compare the "best" clustering in the tree built by the spectral algorithm to the true clustering, where the "best" clustering in the tree is the one that most agrees with the true clustering according to a variety of standard measures – f-measure, entropy, and accuracy. The results show that the spectral algorithm performs better than or competitively with several leading hierarchical clustering algorithms.

The results from Section 4.2 show that a good flat clustering exists in the tree created by the divide phase. In Section 4.3, we give experimental results on the ability of the merge phase to actually *find* this clustering that exists in the tree. We explore how some natural objective functions (k-means, min-sum, min-diameter) perform in practice on real-world data, and compare two flat clusterings: the clustering found by the objective function in the merge phase and the best clustering that exists in the tree. Our results show that the clustering found by the merge phase is only slightly worse than the best possible flat clustering in the tree.

2. DIVIDE-AND-MERGE METHODOLOGY

As mentioned in the introduction, there are two phases in our approach. The divide phase produces a hierarchy and can be implemented using any algorithm that partitions a set into two disjoint subsets. The input to this phase is a set of objects whose pairwise similarities or distances are given, or can be easily computed from the objects themselves. The algorithm recursively partitions a cluster into two smaller sets until it arrives at singletons. The output of this phase is a tree whose leaves are the objects themselves; each internal node represents a subset of the objects, namely the leaves in the subtree below it. Divisive algorithms that can be applied in the divide phase are known for a variety of data representations such as graphs [Dhi01] and high-dimensional vectors [Bol98]. In Section 2.1, we suggest a spectral algorithm analyzed in [KVV04] for the divide phase. We describe an implementation that maintains sparsity of the data when the objects are represented as feature vectors and the similarity between the objects is the inner product between the corresponding vectors.

The merge phase is applied to the tree T produced by the divide phase. The output of the merge phase is a partition $C_1, \ldots, C_k$ of the set of objects and each C_i is a node of T. The merge phase uses a dynamic program to find the optimal tree-respecting clustering for a given objective function g. The optimal solutions are computed bottom-up on T; to compute the optimal solution for any interior node C, we *merge* the optimal solutions for C_l and C_r, the children of C. The optimal solution for any node need not be just a clustering; an optimal solution can be parameterized in a number of ways. Indeed, we can view computing the optimal solution for an interior node as computing a Pareto curve. A value on the curve at a particular point is the optimal solution with the parameters described by the point. A specific objective function g can be efficiently optimized on T if the Pareto curve for a cluster can be efficiently computed from the Pareto curves of its children. The Pareto curve of the root node gives the tradeoff between the parameters and the value of the objective function[1]. The choice of objective function is up to the user and can be tailored to the specific application area. In Section 2.2, we describe dynamic programs to compute optimal tree-respecting clusterings for several well-known objective functions: k-means, min-diameter, min-sum, and correlation clustering.

2.1 A spectral algorithm for the divide phase

In this section, we give an implementation of the spectral algorithm described and analyzed in [KVV04]. The algorithm from [KVV04] takes as input a similarity matrix encoding the similarity between objects and outputs a hierarchical clustering tree. Our implementation deals with the common case where the objects are given as feature vectors, and the similarity between the objects is defined to be the inner product of their feature vectors. Together, the objects form a sparse document-term matrix $A \in \mathbb{R}^{n \times m}$; the rows are the objects and the columns are the features. When A is sparse and n large, it is impractical to apply the spectral algorithm in [KVV04] as a black box. This is because explicitly computing the similarity matrix by computing the inner products takes n^2 space, which can be much larger[2] than the number of non-zeros M in the *document-term* matrix and thus infeasible to store. The implementation we describe in this section takes as input the document-term matrix A and produces a hierarchical clustering tree with the same guarantees as the algorithm from [KVV04]. The key benefit of our implementation is that it uses space linear in M and has a near-linear running time in M.

The algorithm constructs a hierarchical clustering of the objects by recursively dividing a cluster C into two pieces through a cut $(S, C \setminus S)$. To find the cut, we compute v, an approximation of the second eigenvector of the similarity matrix AA^T normalized so that all row sums are 1. The ordering of the coordinates of v gives a set of $n-1$ cuts, and we take the "best" cut (we describe what the "best" cut is in the next paragraph). The algorithm then recurses on the subparts. To compute the approximation of the second eigenvector, we use the power method, a technique

[1]For instance, the tradeoff might be between the number of clusters used and the amount of error incurred – an example will be given for k-means in Section 2.2.

[2]For instance, documents are often described in the bag-of-words model by their top-k distinguishing features, with $k < 500$.

```
**Input:** An $n \times m$ matrix $A$.
**Output:** A tree with the rows of $A$ as leaves.
  (1) Let $\rho \in \mathbb{R}^n$ be a vector of the row sums of $AA^T$, and $\pi = \frac{1}{(\sum_i \rho_i)}\rho$.
  (2) Let $R, D$ be diagonal matrices with $R_{ii} = \rho_i$, $D_{ii} = \sqrt{\pi_i}$.
  (3) Compute the second largest eigenvector $v'$ of $Q = DR^{-1}AA^TD^{-1}$.
  (4) Let $v = D^{-1}v'$, and sort $v$ so that $v_i \leq v_{i+1}$.
  (5) Find $t$ such that the cut
          $(S,T) = (\{1,\ldots,t\},\{t+1,\ldots,n\})$
      minimizes the conductance:
          $\phi(S,T) = \frac{c(S,T)}{\min(c(S),c(T))}$
      where $c(S,T) = \sum_{i\in S, j\in T} A_{(i)} \cdot A_{(j)}$, and $c(S) = C(S,\{1\ldots,n\})$.
  (6) Let $\hat{A}_S$, $\hat{A}_T$ be the submatrices of $A$. Recurse (Steps 1-5) on $\hat{A}_S$ and $\hat{A}_T$.
```

Table I. Divide phase

for which it is not necessary to explicitly compute the normalized similarity matrix AA^T. We describe this in more detail in Section 2.1.1. The algorithm is given in Table I. There, we denote the ith object, a row vector in A, by $A_{(i)}$. The similarity of two objects is defined as the inner product of their term vectors: $A_{(i)} \cdot A_{(j)}$.

In Step 5 of the algorithm, we consider $n-1$ different cuts and use the cut with the smallest conductance. Why should we think that the best cut is a cut of small conductance? Why not just use the minimum cut (i.e. the cut with the minimum weight across it)? Consider Figure 2.1; the nodes are the objects and the edges mean that two objects are very similar. Although both cut C_1 and C_2 have the same number of edges crossing the cut, it is clear that C_1 is a better cut – this is because C_1 partitions the set into two subsets of equal size, both of which have high weight. The measure of conductance formalizes this by normalizing a cut by the smaller weight of the partition it induces. More intuition for why conductance is a good measure for clustering can be found in [KVV04].

The cut (S,T) we find using the second eigenvector in Step 5 is not the cut of *minimum* conductance; finding such a cut is NP-hard. However, the conductance of (S,T) is not much worse than the minimum conductance cut. Sinclair and Jerrum showed that $\phi(S,T) \leq \sqrt{2 \cdot \phi_{\mathsf{OPT}}}$ [SJ89; KVV04].

For a document-term matrix with n objects and M nonzeros, Steps 1-5 take $O(M \log n)$ time. Theoretically, the worst-case time for the spectral algorithm to compute a complete hierarchical clustering of the rows of A is $O(Mn \log n)$; this occurs if each cut the spectral algorithm makes only separates one object from the rest of the objects. Experiments, however, show that the algorithm performs much better (see Section 2.1.2). Indeed, if the spectral algorithm always makes balanced cuts, then the running time for creating a hierarchical clustering is $O(M \log^2 n)$. We discuss this in more detail in Section 2.1.2.

Any vector or matrix that the algorithm uses is stored using standard data structures for sparse representation. The main difficulty is to ensure that the similarity matrix AA^T is not explicitly computed; if it is, we lose sparsity and our running time could grow to n^2. In the next section, we briefly describe how to avoid com-

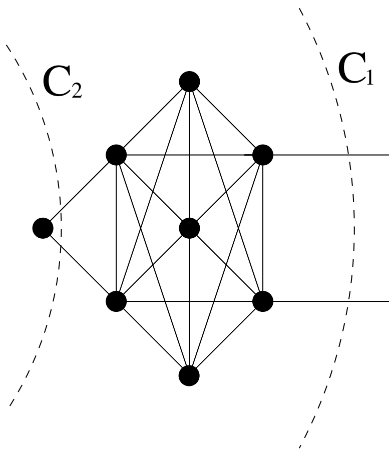


Fig. 2. Minimum conductance cut vs. minimum cut

puting AA^T in Steps 1 and 3. We also describe how to efficiently compute the $n-1$ conductances in Step 5. By avoiding the computation of AA^T, the algorithm runs in space $O(M)$.

2.1.1 *Details of the spectral algorithm.*

2.1.1.1 *Step 1: Computing row sums..* Observe that

$$\rho_i = \sum_{j=1}^{n} A_{(i)} \cdot A_{(j)} = \sum_{j=1}^{n}\sum_{k=1}^{m} A_{ik}A_{jk} = \sum_{k=1}^{m} A_{ik}\left(\sum_{j=1}^{n} A_{jk}\right).$$

Because $\sum_{j=1}^{n} A_{jk}$ does not depend on i, we can compute $u = \sum_{i=1}^{n} A_{(i)}$ so we have that $\rho_i = A_{(i)} \cdot u$. The total running time is $O(M)$ and the additional space required is $O(n+m)$.

2.1.1.2 *Step 3: Computing the eigenvector..* The algorithm described in [KVV04] uses the second largest eigenvector of $B = R^{-1}AA^T$, the normalized similarity matrix, to compute a good cut. To compute this vector efficiently, we compute the second largest eigenvector v of the matrix $Q = DBD^{-1}$. The eigenvectors and eigenvalues of Q and B are related – if v is such that $Bv = \lambda v$, then $Q(Dv) = \lambda Dv$.

The key property of Q is that it is a symmetric matrix. It is easy to see this from the fact that $D^2B = B^TD^2$ and D is a diagonal matrix (so $D^T = D$):

$$D^2B = B^TD^2 \rightarrow D^{-1}D^2B = D^{-1}B^TD^2 \rightarrow D^{-1}D^2BD^{-1} = D^{-1}B^TD^2D^{-1} \rightarrow Q = Q^T.$$

Since Q is symmetric, we can compute the second largest eigenvector of Q using the power method, an iterative algorithm whose main computation is a matrix-vector multiplication.

Power Method

(1) Let $v \in \mathbb{R}^n$ be a random vector orthogonal to π^TD^{-1}.

(2) Repeat
   (a) Normalize v, i.e. set $v = v/||v||$.
   (b) Set $v = Qv$.

Step 1 ensures that the vector we compute is the second largest eigenvector. Note that $\pi^T D^{-1} Q = \pi^T D^{-1}$, so πD^{-1} is a left eigenvector with eigenvalue 1. To evaluate Qv in Step 2, we only need to do four sparse matrix-vector multiplications ($v := D^{-1}v$, followed by $v := A^T v$, $v := Av$, and $v := DR^{-1}v$) since $Q = (DR^{-1}AA^T D^{-1})$, and each of these matrices is sparse. Therefore, the space used is $O(M)$, linear in the number of nonzeros in the document-term matrix A.

The following lemma shows that the power method takes $O(\log n)$ iterations to converge to the top eigenvector. Although stated for the top eigenvector, the lemma and theorem still hold when the starting vector is chosen uniformly over vectors orthogonal to the top eigenvector $\pi^T D^{-1}$; in this case, the power method will converge to the second largest eigenvector (since the second eigenvector is orthogonal to the first). The analysis of the power method is standard and classical (see e.g. [GL96]). Our analysis differs in two respects. First, the classical analysis assumes that $|\lambda_1| > |\lambda_2|$ – we do not need the assumption because if $\lambda_1 = \lambda_2$, the cut we find partitions the graph into two pieces with no edges crossing the cut. Second, the classical analysis states convergence in terms of the size of the projection of the starting vector on the first eigenvector; in our analysis, we quantify how large this is for a random vector.

LEMMA 1. *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix, and let $v \in \mathbb{R}^n$ be chosen uniformly at random from the unit n-dimensional sphere. Then for any positive integer k, the following holds with probability at least $1-\delta$:*

$$\frac{||A^{k+1}v||}{||A^k v||} \geq \left(n \ln \frac{1}{\delta}\right)^{-\frac{1}{2k}} ||A||_2.$$

PROOF. Since A is symmetric, we can write

$$A = \sum_{i=1}^{n} \lambda_i u_i u_i^T,$$

where the λ_i 's are the eigenvalues of A arranged in the order $|\lambda_1| \geq |\lambda_2| \dots |\lambda_n|$ and the u_i are the corresponding eigenvectors. Note that, by definition, $\lambda_1 = ||A||_2$. Express v in this basis as $v = \sum_i \alpha_i u_i$, where $\sum_i \alpha_i^2 = 1$. Since, v is uniformly random over the unit-dimensional sphere, we have that with probability at least $1-\delta$, $\alpha_1^2 \geq 1/(n \ln(1/\delta))$. It is easy to see that, in expectation, $\alpha_1^2 = 1/n$ – this follows from the symmetry of the sphere. The tail bound follows from the fact that the distribution of the projection of a point from the sphere to a random line behaves roughly like a Gaussian random variable. Then, using Hölder's inequality (which says that for any $p, q > 0$ satisfying $(1/p) + (1/q) = 1$ and any $a, b \in \mathbb{R}^n$, we have $\sum_i a_i b_i \leq (\sum_i |a_i|^p)^{1/p} (\sum_i |b_i|^q)^{1/q}$), we have

$$||A^k v||^2 = \sum_i \alpha_i^2 \lambda_i^{2k} \leq \left(\sum \alpha_i^2 \lambda_i^{2k+2}\right)^{k/(k+1)}$$

where the last inequality holds using Hölder with $p = 1 + (1/k)$ $\quad q = k + 1$ $\quad a_i = \alpha_i^{2k/(k+1)}\lambda_i^{2k}$ $\quad b_i = \alpha_i^{2/(k+1)}$. Note that $||A^{k+1}v|| = \sum_i \alpha_i^2\lambda_i^{2k+2}$. Combining this with the previous inequality we have:

$$\frac{||A^{k+1}v||}{||A^k v||} \geq \frac{\sum_i \alpha_i^2\lambda_i^{2k+2}}{\left(\sum \alpha_i^2\lambda_i^{2k+2}\right)^{k/(k+1)}} \geq \left(\sum \alpha_i^2\lambda_i^{2k+2}\right)^{1/(k+1)} \geq \left(\alpha_1^2\lambda_1^{2k+2}\right)^{1/(k+1)}.$$

As concluded above, with probability $1-\delta$, $\alpha_1^2 \geq 1/(n\ln(1/\delta))$. This gives us the desired result. □

The following corollary quantifies the number of steps to run the power method to find a good approximation.

Corollary 1. *If* $k \geq \frac{1}{2\epsilon}\ln(n\ln(\frac{1}{\delta}))$, *then we have:*

$$\frac{||A^{k+1}v||}{||A^k v||} \geq (1-\epsilon)\lambda_1.$$

2.1.1.3 *Step 5: Computing conductance of* $n-1$ *cuts..* We choose the cut C of the $n-1$ cuts $(\{1,\ldots,t\},\{t+1,\ldots,n\})$ which has the smallest conductance. Recall that

$$\phi(\{1,\ldots,i\},\{i+1,\ldots,n\}) = \frac{\sum_{k=1}^{i}\sum_{j=i+1}^{n} A_{(k)}\cdot A_{(j)}}{\min(\sum_{k=1}^{i}\rho_k, \sum_{k=i+1}^{n}\rho_k)}.$$

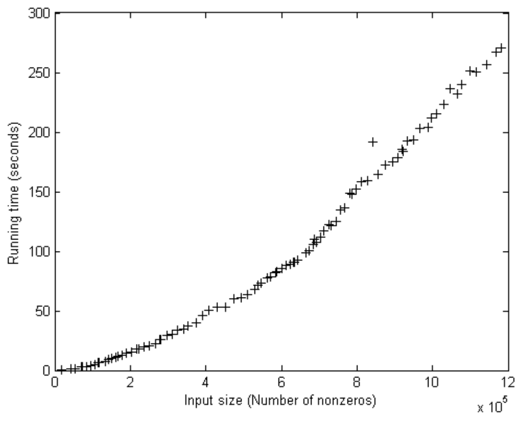
Let the numerator of this expression be u_i, and the denominator be l_i.

We can compute u_i from u_{i-1} as follows. Let $x_i = A_{(1)} + \ldots + A_{(i)}$ and $y_i = A_{(i+1)} + \ldots + A_{(n)}$. Then $u_1 = x_1 \cdot y_1$, and

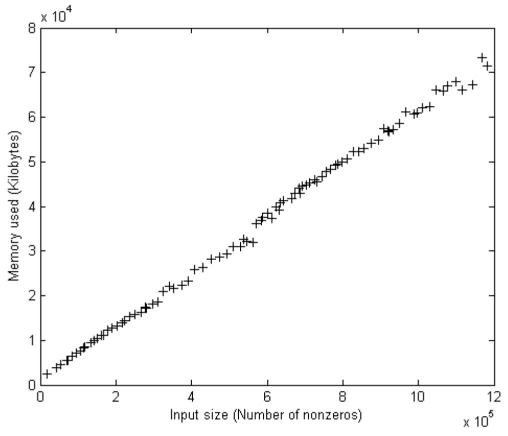
$$u_i = (x_{i-1} + A_{(i)})\cdot(y_{i-1} - A_{(i)}) = u_{i-1} - x_{i-1}\cdot A_{(i)} + y_{i-1}\cdot A_{(i)} + A_{(i)}\cdot A_{(i)}.$$

The denominator, l_i can be computed in a similar fashion. Since we only require one pass through A to compute the values of these $n-1$ cuts, the time and space used is $O(M)$.

2.1.2 *Time and space requirements.* In practice, the spectral algorithm does not run in the worst-case $O(Mn\log n)$ time. If each cut made by the algorithm is balanced, then the spectral algorithm runs in $O(M\log^2 n)$ time. By a balanced cut, we mean that both the number of nonzeros and number of rows on the larger side of the cut are at most a constant fraction (say, 2/3) of the total number of nonzeros and rows, respectively. If each cut is balanced, the depth of the hierarchical clustering tree is at most $O(\log n)$. Since the running time at each level of the tree is $M\log n$, the total running time when each cut is balanced is bounded above by $O(M\log^2 n)$. On real world data, the algorithm seems to run in time roughly $O(M\log^5 n)$. Figures 3(a) and 3(b) show the results of a performance experiment. In this experiment, we computed a complete hierarchical clustering for N newsgroup articles in the 20 newsgroups dataset [Lan] and measured the running time and memory used. The value of N ranged from 200 to 18,000. When we clustered $18,000$ documents (for a total of 1.2 million nonzeros in the document-term matrix), we were able to compute a complete hierarchical clustering in 4.5 minutes on



(a) Time vs. input size



(b) Space vs. input size

Fig. 3. Performance of spectral algorithm in experiments

commodity hardware (a 3.2 Ghz Pentium IV with 1 gigabyte of RAM). Note that the space used is linear in the size of the input.

In some applications, knowledge about the dataset can be used to halt the spectral algorithm before a complete tree is constructed. For instance, if the number of clusters k desired is small, the recursive step does not need to be applied after depth k, since all k-clusterings in the tree use nodes above depth k. Here is why – if a node t at depth $k+1$ is a cluster, then no node along the path from t to the root is also a cluster. Since each node along this path has two children, and each leaf node must be covered by an interior node, there are at least $k+1$ other nodes that need to be covered by distinct clusters, contradicting the use of only k clusters.

2.2 Merge phase

The merge phase finds an optimal clustering in the tree produced by the divide phase. Recall that the tree produced by the divide phase has two properties: (1) each node in the tree is a subset of the objects, (2) the left and right children of a node form a partition of the parent subset. A *clustering in the tree* is thus a subset S of nodes in the tree such that each leaf node is "covered", i.e. the path from a leaf to root encounters exactly one node in S.

In this section, we give dynamic programs to compute the optimal clustering in the tree for many standard objective functions. If we are trying to maximize the objective function g, the dynamic program will find a clustering $C_{\textsf{OPT-TREE}}$ in the tree such that $g(C_{\textsf{OPT-TREE}}) \geq g(C)$, for any clustering C in the tree. Note that the best clustering in the tree may not be the best possible clustering. Indeed, the best possible clustering may not respect the tree. In practice, we have found that good clusterings do exist in the tree created by the spectral algorithm and that the merge phase, with appropriate objective functions, finds these clusterings (see Section 4.3).

In general, the running time of the merge phase depends on both the number of times we must compute the objective function and the evaluation time of the objective function itself. Suppose at each interior node we compute a Pareto curve of k points from the Pareto curves of the node's children. Let c be the cost of evaluating the objective function. Then the total running time is $O(nk^2 + nkc)$: linear in n and c, with a small polynomial dependence on k.

k-means: The k-means objective function seeks to find a k-clustering such that the sum of the squared distances of the points in each cluster to the centroid p_i of the cluster is minimized:

$$g(\{C_1, \ldots, C_k\}) = \sum_i \sum_{u \in C_i} d(u, p_i)^2.$$

The centroid of a cluster is just the average of the points in the cluster. This problem is NP-hard; several heuristics (such as the k-means *algorithm*) and approximation algorithms exist (e.g. [HW79; KSS04]). Let $\textsf{OPT-TREE}(C, i)$ be the optimal tree-respecting clustering for C using i clusters. Let C_l and C_r be the left and right children of C in T. Then we have the following recurrence:

$$\textsf{OPT-TREE}(C, 1) = \{C\}$$

since we are constrained to only use 1 cluster. When $i > 1$, we have:

$$\textsf{OPT-TREE}(C, i) = \textsf{OPT-TREE}(C_l, j) \cup \textsf{OPT-TREE}(C_r, i - j)$$

where

$$j = \text{argmin}_{1 \leq j < i}\ g(\textsf{OPT-TREE}(C_l, j) \cup \textsf{OPT-TREE}(C_r, i - j)).$$

By computing the optimal clustering for the leaf nodes first, we can determine the optimal clustering efficiently for any interior node. Then $\textsf{OPT-TREE}(\textsf{root}, k)$ gives the optimal clustering. Note that in the process of finding the optimal clustering the dynamic program finds the Pareto curve $\textsf{OPT-TREE}(\textsf{root}, \cdot)$; the curve describes the tradeoff between the number of clusters used and the "error" incurred.

Min-diameter: We wish to find a k-clustering for which the cluster with maximum diameter is minimized:

$$g(\{C_1, \ldots, C_k\}) = \max_i \mathsf{diam}(C_i).$$

The diameter of any cluster is the maximum distance between any pair of objects in the cluster. A similar dynamic program to that above can find the optimal tree-respecting clustering. This objective function has been investigated in [CFM97].

Min-sum: Another objective considered in the literature is minimizing the sum of pairwise distances within each cluster:

$$g(\{C_1, \ldots, C_k\}) = \sum_{i=1}^{k} \sum_{u,v \in C_i} d(u,v).$$

We can compute an optimal tree-respecting clustering in the tree T by a similar dynamic program to the one above. Although approximation algorithms are known for this problem (as well as the one above), their running times seem too large to be useful in practice [dlVKKR03].

Correlation clustering: Suppose we are given a graph where each pair of vertices is either deemed similar (red) or not (blue). Let R and B be the set of red and blue edges, respectively. Correlation clustering seeks to find a partition that maximizes the number of agreements between a clustering and the edges – i.e. maximizing the number of red edges within clusters plus the number of blue edges between clusters:

$$g(\{C_1 \ldots C_k\}) \; = \; \sum_i |\{(u,v) \in R \cap C_i\}| + \frac{1}{2}|\{(u,v) \in B : u \in C_i, v \in U \setminus C_i\}|.$$

Let C be a cluster in the tree T, and let C_l and C_r be its two children. The dynamic programming recurrence for $\mathsf{OPT\text{-}TREE}(C)$ is:

$$\mathsf{OPT\text{-}TREE}(C) = \text{argmax } \{g(C), g(\mathsf{OPT\text{-}TREE}(C_l) \cup \mathsf{OPT\text{-}TREE}(C_r)).$$

If, instead, we are given pairwise similarities in $[0,1]$, where 0 means dissimilar and 1 means similar, we can define two thresholds t_1 and t_2. Edges with similarity greater than t_1 are colored red and edges with similarity less than t_2 are colored blue. The same objective function can be applied to these new sets of edges $R_{(t_1)}$ and $B_{(t_2)}$. Approximation algorithms have been given for this problem as well, although the techniques used (linear and semidefinite programming) incur large computational overhead [BBC02; CGW03; DI03; EF03; Swa04].

2.3 Choice of algorithms for divide, merge steps

We have suggested a spectral algorithm for the divide phase and several different objective functions for the merge phase. A natural question is: how do these two phases interact, i.e. how does the choice of the algorithm for the divide phase affect the performance of the merge phase?

A natural approach is to use the same objective function for the divide phase as the merge phase – that is, recursively find the optimal 2-clustering according to the objective function. This process results in a hierarchical clustering tree. In the merge phase, use the same objective function to find the best clustering. In practice, there can be difficulty with this approach because finding the optimal 2-clustering can be NP-hard as well (for instance, for the k-means, k-median objective functions). Even if we could find the optimal 2-clustering, the hierarchical clustering tree is not necessarily good for all k. This is the case for the min-diameter objective function, where we seek a clustering that minimizes the maximum diameter between two points in the same cluster. Consider the following four points on the real number line: $0, 1/2 - \epsilon, 1/2 + \epsilon, 1$. The best 2-clustering is $\{0, 1/2 - \epsilon\}, \{1/2 + \epsilon, 1\}$ with a maximum radius of $1/2 - \epsilon$. However, the optimal 3-clustering is $\{0\}, \{1/2 - \epsilon, 1/2 + \epsilon\}, \{1\}$ with a maximum radius of 2ϵ, which does not respect the initial partition. The best 3-clustering *in the tree* has maximum radius $1/2 - \epsilon$, so the ratio of the best 3-clustering in the tree to the optimal 3-clustering cannot be bounded by a constant. The situation is better for other objective functions. In the min k-cut objective function, we seek a k-clustering where the sum of the pairwise similarities across clusters is minimized. Saran and Vazirani show that the k-clustering obtained by greedily cutting the subset with the smallest min-cut k times is a factor $2 - 2/k$ approximation. The tree formed by recursively finding the min-cut includes this greedy k-clustering, which gives the same factor $2 - 2/k$ approximation guarantee for the divide-and-merge methodology.

3. APPLICATION TO STRUCTURING WEB SEARCHES: EIGENCLUSTER

In a standard web search engine, the results for a given query are ranked in a linear order. Although suitable for some queries, the linear order fails to show the inherent clustered structure of results for queries with multiple meanings. For instance, consider the query `mickey`. The query can refer to multiple people (Mickey Rooney, Mickey Mantle) or the fictional character Mickey Mouse.

We have implemented the divide-and-merge methodology in a meta-search engine that discovers the clustered structure for queries and identifies each cluster by its three most significant terms. The website is located at `http://eigencluster.csail.mit.edu`. The user inputs a query which is then used to find 400 results from Google, a standard search engine. Each result contains the title of the webpage, its location, and a small snippet from the text of the webpage. We construct a document-term matrix representation of the results; each result is a document and the words in its title and snippet make up its terms. Standard text pre-processing such as TF/IDF, removal of stopwords, and removal of too frequent or infrequent terms is applied [VR79]. The similarity between two results is the inner product between their two term vectors.

The divide phase was implemented using our spectral algorithm. For the merge phase, we used the following objective function, which we refer to as relaxed correlation clustering:

$$\sum_i \alpha \left(\sum_{u,v \in C_i} 1 - A_{(u)} \cdot A_{(v)} \right) + \beta \left(\sum_{u \in C_i, v \notin C_i} A_{(u)} \cdot A_{(v)} \right).$$

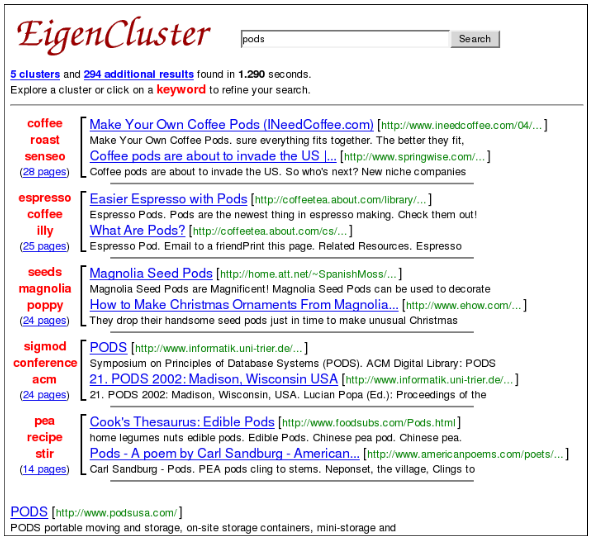
We assume here that each row $A_{(u)}$ is normalized to have Euclidean length 1; this is a standard preprocessing step that ensures that the maximum similarity between any pair of rows is 1. In EigenCluster, we use $\alpha = .2$, and $\beta = .8$. The first term, $\alpha \sum_{u,v \in C_i}(1 - A_{(u)} \cdot A_{(v)})$ measures the dissimilarity within a cluster, i.e. how "far" a cluster is from a set in which every pair is as similar as possible (for all u, v, $A_{(u)} \cdot A_{(v)} = 1$). Note that the first term is a relaxed notion of the blue edges within a cluster from correlation clustering. The second term, $\beta \sum_{u \in C_i, v \notin C_i} A_{(u)} \cdot A_{(v)}$ measures the amount of similarity the clustering fails to capture, since it occurs across clusters. Similarly, the second term is a relaxed notion of the red edges outside clusters. The benefit of using the relaxed correlation clustering objective function is that it does not depend on a predefined number of clusters k. This is appropriate for our application, since the number of meanings or contexts of a query could not possibly be known beforehand. We have seen in practice that the objective function does a good job of picking out the large, interesting subsets of the data while putting unrelated results each in their own cluster.

Sample queries can be seen in Figures 4(a) and 4(c); in each example, EigenCluster identifies the multiple meanings of the query as well as keywords corresponding to those meanings. Furthermore, many results are correctly labeled as singletons. Figures 4(a) and 4(c) show screenshots of EigenCluster and Figures 4(b) and 4(d) are before and after depictions of the similarity matrix. The (i, j)th entry of the matrix represents the similarity between results i and j – the darker the pixel, the more similar i and j are. In the before picture, the results are arranged in the order received from Google. In the after picture, the results are arranged according to the cuts made by the spectral algorithm. The cluster structure is apparent. EigenCluster takes roughly 0.7 seconds to fetch and cluster results on a Pentium III 700 megahertz with 512 megabytes of RAM. Section 6 shows more example EigenCluster searches.

4. COMPARATIVE EXPERIMENTS ON STANDARD DATASETS

In this section, we conduct a thorough experimental evaluation of the divide-and-merge methodology. We work with real-world datasets (text, gene expression, and categorical data) for which a labeling of data objects is known. In Section 4.2, we apply the spectral algorithm as a divide phase to the data. The results show that the tree the spectral algorithm constructs is good – there exists a clustering within the tree that "agrees" with the true clustering, i.e. the partition of data objects into sets whose members share the same label. This type of evaluation in which a clustering algorithm is applied to data for which the true classification is known is common. It is an appropriate evaluation because the true classification reflects the *true structure* of the data. Comparing the clustering found by the algorithm to the true classification measures the ability of a clustering algorithm to find true structure in data, which is a primary use of clustering in practice. The results of our experiments compare favorably to results obtained from leading hierarchical clustering algorithms.

In Section 4.3, we proceed to experimentally evaluate the merge phase. We evaluate how each of the objective functions k-means, min-sum, and min-diameter behave on the tree constructed by the spectral algorithm. We find that the merge



EigenCluster

pods | Search

5 clusters and 294 additional results found in **1.290** seconds.
Explore a cluster or click on a **keyword** to refine your search.

coffee roast senseo (28 pages)
- Make Your Own Coffee Pods (INeedCoffee.com) [http://www.ineedcoffee.com/04/...]
  Make Your Own Coffee Pods. sure everything fits together. The better they fit,
- Coffee pods are about to invade the US |... [http://www.springwise.com/...]
  Coffee pods are about to invade the US. So who's next? New niche companies

espresso coffee illy (25 pages)
- Easier Espresso with Pods [http://coffeetea.about.com/library/...]
  Espresso Pods. Pods are the newest thing in espresso making. Check them out!
- What Are Pods? [http://coffeetea.about.com/cs/...]
  Espresso Pod. Email to a friendPrint this page. Related Resources. Espresso

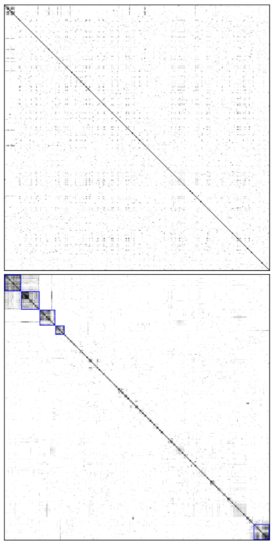
seeds magnolia poppy (24 pages)
- Magnolia Seed Pods [http://home.att.net/~SpanishMoss/...]
  Magnolia Seed Pods are Magnificent! Magnolia Seed Pods can be used to decorate
- How to Make Christmas Ornaments From Magnolia... [http://www.ehow.com/...]
  They drop their handsome seed pods just in time to make unusual Christmas

sigmod conference acm (24 pages)
- PODS [http://www.informatik.uni-trier.de/...]
  Symposium on Principles of Database Systems (PODS). ACM Digital Library: PODS
- 21. PODS 2002: Madison, Wisconsin USA [http://www.informatik.uni-trier.de/...]
  21. PODS 2002: Madison, Wisconsin, USA. Lucian Popa (Ed.): Proceedings of the

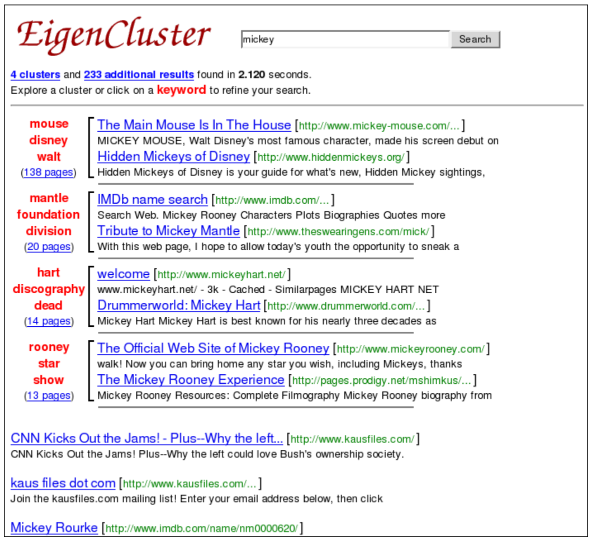
pea recipe stir (14 pages)
- Cook's Thesaurus: Edible Pods [http://www.foodsubs.com/Pods.html]
  home legumes nuts edible pods. Edible Pods. Chinese pea pod. Chinese pea.
- Pods - A poem by Carl Sandburg - American... [http://www.americanpoems.com/poets/...]
  Carl Sandburg - Pods. PEA pods cling to stems. Neponset, the village, Clings to

PODS [http://www.podsusa.com/]
PODS portable moving and storage, on-site storage containers, mini-storage and

(a) Query: pods



(b) Before/after: pods



EigenCluster

mickey | Search

4 clusters and 233 additional results found in **2.120** seconds.
Explore a cluster or click on a **keyword** to refine your search.

mouse disney walt (138 pages)
- The Main Mouse Is In The House [http://www.mickey-mouse.com/...]
  MICKEY MOUSE, Walt Disney's most famous character, made his screen debut on
- Hidden Mickeys of Disney [http://www.hiddenmickeys.org/]
  Hidden Mickeys of Disney is your guide for what's new, Hidden Mickey sightings,

mantle foundation division (20 pages)
- IMDb name search [http://www.imdb.com/...]
  Search Web. Mickey Rooney Characters Plots Biographies Quotes more
- Tribute to Mickey Mantle [http://www.theswearingens.com/mick/]
  With this web page, I hope to allow today's youth the opportunity to sneak a

hart discography dead (14 pages)
- welcome [http://www.mickeyhart.net/]
  www.mickeyhart.net/ - 3k - Cached - Similarpages MICKEY HART NET
- Drummerworld: Mickey Hart [http://www.drummerworld.com/...]
  Mickey Hart Mickey Hart is best known for his nearly three decades as

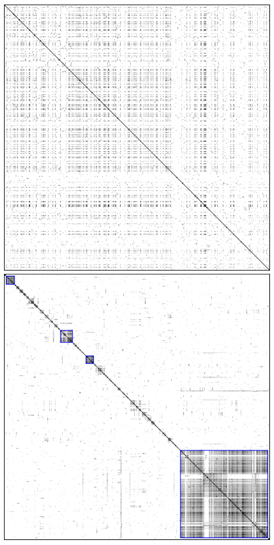
rooney star show (13 pages)
- The Official Web Site of Mickey Rooney [http://www.mickeyrooney.com/]
  walk! Now you can bring home any star you wish, including Mickeys, thanks
- The Mickey Rooney Experience [http://pages.prodigy.net/mshimkus/...]
  Mickey Rooney Resources: Complete Filmography Mickey Rooney biography from

CNN Kicks Out the Jams! - Plus--Why the left... [http://www.kausfiles.com/]
CNN Kicks Out the Jams! Plus--Why the left could love Bush's ownership society.

kaus files dot com [http://www.kausfiles.com/...]
Join the kausfiles.com mailing list! Enter your email address below, then click

Mickey Rourke [http://www.imdb.com/name/nm0000620/]

(c) Query: mickey



(d) Before/after: mickey

Fig. 4. Example EigenCluster searches

phase can indeed find a good clustering; it typically finds a clustering only slightly worse than the "best" clustering that exists in the tree. The best clustering in the tree is the one that most closely matches the true clustering.

The next section describes how we compare a clustering (either hierarchical or flat) to the true clustering.

4.1 Comparative measures

Let the true classification of a dataset be $C_1, \ldots, C_k$. We refer to each C_i as a *class*. Let $\hat{C}_1, \ldots, \hat{C}_l$ be subsets of the universe $U = \bigcup_i C_i$. Note that $\hat{C}_1, \ldots, \hat{C}_l$ may have a non-empty intersection (for instance, when each $\hat{C}_l$ is the set of leaves underneath a node of a hierarchical clustering tree). We will note when $\hat{C}_1, \ldots, \hat{C}_l$ is partition of U, rather than just a collection of subsets.

F-measure: For each class C_i, the F-measure of that class is:

$$F(i) = \max_{j=1}^{l} \frac{2P_jR_j}{P_j + R_j}$$

where:

$$P_j = \frac{|C_i \cap \hat{C}_j|}{|\hat{C}_i|}, R_j = \frac{|C_i \cap \hat{C}_j|}{|C_i|}$$

P_j is referred to as *precision* and R_j is referred to as *recall*. The F-measure of the clustering is defined as:

$$\sum_{i=1}^{k} F(i) \cdot \frac{|C_i|}{|C|}.$$

The F-measure score is in the range $[0, 1]$ and a *higher* F-measure score implies a better clustering. Note that the F-measure does not assume that $\hat{C}_1, \ldots, \hat{C}_l$ is a partition of U; indeed, it is often used to compare a hierarchical clustering to a true classification. For a more in-depth introduction and justification to the F-measure, see e.g. [VR79; LA99; BEX02; NJM01].

Entropy: We consider $\hat{C}_1, \ldots, \hat{C}_k$ to be a partition of U. For each $\hat{C}_j$, we define the entropy of $\hat{C}_j$ as:

$$E(\hat{C}_j) = \sum_{i=1}^{k} - \left(\frac{|C_i \cap \hat{C}_j|}{|\hat{C}_j|} \right) \log \left(\frac{|C_i \cap \hat{C}_j|}{|\hat{C}_j|} \right)$$

The entropy of a subset is a measure of the disorder within the cluster. As such, a *lower* entropy score implies that a clustering is better; the best possible entropy score is 0. Entropy was first introduced in [Sha48] and has been used as a measure of clustering quality in [Bol98; Dhi01; BLC02].

The entropy of a partition $\hat{C}_1 \ldots \hat{C}_k$ is the weighted sum of the entropies of the clusters.

Accuracy: The accuracy of a partition $\hat{C}_1, \ldots, \hat{C}_k$ is defined as:

$$\max_{\pi \in S_k} \frac{\sum_{i=1}^{k} |C_i \cap \hat{C}_{\pi(i)}|}{|U|}$$

where S_k is the set of all permutations on k items. Note that the range of an accuracy score is between 0 and 1; the *higher* the accuracy score, the better. Accuracy, which has been used as a measure of performance in supervised learning, has also

dataset	Spectral	p-QR	p-Kmeans	K-means
alt.atheism comp.graphics	93.6 ± 2.6	89.3 ± 7.5	89.6 ± 6.9	76.3 ± 13.1
comp.graphics comp.os.ms-windows.misc	81.9 ± 6.3	62.4 ± 8.4	63.8 ± 8.7	61.6 ± 8.0
rec.autos rec.motorcycles	80.3 ± 8.4	75.9 ± 8.9	77.6 ± 9.0	65.7 ± 9.3
rec.sport.baseball rec.sport.hockey	70.1 ± 8.9	73.3 ± 9.1	74.9 ± 8.9	62.0 ± 8.6
alt.atheism sci.space	94.3 ± 4.6	73.7 ± 9.1	74.9 ± 8.9	62.0 ± 8.6
talk.politics.mideast talk.politics.misc	69.3 ± 11.8	63.9 ± 6.1	64.0 ± 7.2	64.9 ± 8.5

Table II. 20 Newsgroups dataset (Accuracy)

been used in clustering ([ST00], [ZDG$^+$01]).

Confusion matrix: The confusion matrix for a partition $\hat{C}_1, \ldots, \hat{C}_k$ shows the distribution of the class of the objects in each $\hat{C}_i$ – it is a $k \times k$ matrix M where the rows are the clusters $\hat{C}_i$, and the columns are the classes C_j. The entry M_{ij} denotes the number of objects in $\hat{C}_i$ that belong to class C_j. The order of the clusters $\hat{C}_i$ is chosen so as to maximize the number of elements on the diagonal.[3]

4.2 Spectral algorithm as the divide phase

We tested the spectral algorithm on three types of data: text, gene expression, and categorical data. In all experiments, we compare better than or favorably with the known results. In each of the experiments, the known results come directly as reported in the pertinent paper. To be precise, for each experiment, we ran the same experiment as described in the pertinent paper, but with the spectral algorithm instead of the algorithm given in the paper. In particular, we did not try to validate the findings of each paper by rerunning the experiment with the algorithm given in the paper.

4.2.1 *Text data.*

4.2.1.1 *20 Newsgroups:.* The 20 newsgroups resource [Lan] is a corpus of roughly 20,000 articles that come from 20 specific Usenet newsgroups. We performed a subset of the experiments in [ZDG$^+$01]. Each experiment involved choosing 50 random newsgroup articles each from two newsgroups, constructing term vectors for them, and then applying the spectral algorithm to the document-term matrix.[4] The term vectors were constructed exactly as in [ZDG$^+$01]: words were stemmed,

[3]Maximizing the number of elements on the diagonal can be done via solving a maximum matching problem.
[4]We used the BOW toolkit for processing the newsgroup data. More information on the BOW toolkit can be found on `http://www-2.cs.cmu.edu/~mccallum/bow`.

words that appear too few times were removed, and the TF/IDF weighting scheme was applied.

Since each experiment involved clustering documents from only two classes, we did not need to form a complete hierarchical tree. The first cut made by the spectral algorithm defines a partition into two clusters. Zha et al. also form two clusters using their clustering algorithm. The results can be seen in Table II. Note that we perform better than p-QR, the algorithm proposed in [ZDG$^+$01] on all but one of the experiments. We also outperform K-means and a variation of the K-means algorithm, p-Kmeans. In each of these experiments, the measure of performance was accuracy. Since the experiment involved choosing 50 random newsgroup articles, the experiment was run 100 times and the mean and standard deviation of the results were recorded.

4.2.1.2 *Reuters:.* The Reuters dataset [Lew] is a corpus of 21,578 news articles. Of these, 8,654 articles are uniquely classified into 65 distinct news topics. Previous clustering experiments on this dataset have been conducted by [BEX02; LA99; NJM01]. We performed the same experiments (in particular, forming the term vectors exactly as the experiments specify). In each experiment, a hierarchical clustering was formed, and the F-measure was computed. We perform better than the previous experiments; results appear in Table III. We briefly describe each experiment below.

—In [BEX02], random subsets of size 4,000 of all 8,654 uniquely classified articles were clustered using a hierarchical clustering algorithm. The term vector for each article was constructed by removing stopwords and stemming words. A comparison of our results and their results for this experiment can be found in the first column of Table III.

—Larsen and Aone [LA99] apply a hierarchical clustering algorithm to all 8,654 articles. To form a term vector from an article, they first remove stopwords and apply TF/IDF weighting. The term vector consists of the remaining top 500 highest weight terms in each article. The second column of Table III shows a comparison of our results and their results for this experiment.

—In [NJM01], a hierarchical clustering algorithm was applied to 6,575 of the 8,654 uniquely classified news articles. Each of these 6,575 articles is labeled with one of the following ten labels: earn, acq, money-fx, grain, crude, trade, interest, wheat, ship, or corn. The articles were first preprocessed by removing stopwords and applying Porter's stemming algorithm. The term vector for each document was formed by the counts of the 500 most frequently occurring words. Their results and our results for this experiment are compared in the third column of Table III.

The results of these experiments are summarized below.

	BEX '02	LA '99	NJM '01
Previous	0.49	0.62	0.67
Spectral	0.62	0.75	0.68

Table III. Reuters data (F-measure)

4.2.1.3 *Web pages:.* Boley [Bol98] performs a series of experiments on clustering 185 webpages that fall into 10 distinct categories. In each of the 11 experiments (J1-J11), the term vector for each webpage was constructed in a slightly different way (the exact details can be found in [Bol98][5].). The algorithm from [Bol98] is also a partitional algorithm that constructs a hierarchical clustering. In each experiment, the quality of the clustering is measured by computing the entropy of the 16 clusters at depth 4 in the tree. We measured the entropy of the 16 clusters at depth 4 in our tree as well as in an optimal partition into 16 clusters, allowing clusters at different depths. By an optimal partition into 16 clusters, we mean the 16-clustering in the tree that minimizes entropy. The results from [Bol98] appear in Table IV in the row "Boley '98" and our results appear in the rows labeled "Fixed depth Spectral" (16 clusters at depth 4) and "Optimal Spectral" (optimal 16-clustering in the tree).

The results in Table IV show that the 16 clusters at depth 4 in our tree perform better than the 16 clusters from [Bol98] in all but two experiments. The entropy of the best 16-clustering in the tree does markedly better than both "Fixed depth Spectral" and [Bol98]. This shows that a good clustering exists in the tree; we will see that the merge phase can find a clustering *almost* as good as this in Section 4.3.

	J1	**J2**	**J3**	**J4**	**J5**	**J6**	**J7**	**J8**	**J9**	**J10**	**J11**
Boley '98	0.69	1.12	0.85	1.10	0.74	0.83	0.90	0.96	1.07	1.17	1.05
Fixed depth Spectral	0.77	0.92	0.72	0.94	0.76	0.72	0.84	0.88	0.89	1.04	0.88
Optimal Spectral	0.71	0.63	0.62	0.62	0.71	0.61	0.65	0.63	0.69	0.55	0.83

Table IV. Web page results (Entropy)

4.2.1.4 *SMART dataset:.* The SMART dataset is a set of abstracts originating from Cornell University [Sal] that have been used extensively in information retrieval experiments. The makeup of the abstracts is: 1,033 medical abstracts (Medline), 1,400 aeronautical systems abstracts (Cranfield), and 1,460 information retrieval abstracts (Cisi). The term vector for each abstract was formed by removing stopwords and words that occur in less than 0.2% or greater than 15% of the abstracts.

We performed the same four experiments as those found in [Dhi01]. In the first three experiments, the datasets were the mixture of abstracts from two classes. In the fourth experiment, the dataset was the set of all abstracts. In the first three experiments, we just apply the spectral algorithm once to obtain a 2-clustering of the data set. In the fourth experiment, we recurse with the spectral algorithm twice, and select the better of the two 3-clusterings in the tree.

The results from performing the same experiments are listed in the column labeled "Spectral" in Table V. We do much worse than [Dhi01]. The reason for this is so many terms are removed in the construction of each term vector. As such, the similarity (inner product between two term vectors) may be very small, and the best first conductance cut may separate just one or two objects from the rest

[5]The raw data can be found from `ftp://ftp.cs.umn.edu/dept/users/boley/PDDPdata`

dataset	Spectral (TF/IDF)	Spectral	Dhillon '01
MedCran	0.0172	0.027	0.026
MedCisi	0.0365	0.054	0.152
CisiCran	0.0426	0.490	0.046
Classic3	0.0560	0.435	0.089

Table V. SMART dataset (Entropy)

of the set. While the first cut may not separate the classes, we have found that one of the next cuts often does separate the classes. When we applied TF/IDF weighting in the construction of term vectors, we found much better performance (see the column labeled "Spectral (TF/IDF)" in Table V). Indeed, with TF/IDF term weighting, we perform better than [Dhi01]. The TF/IDF weighting increases the minimum similarity between any two abstracts, so that the best first conductance cut does not separate just one or two objects from the set.

4.2.2 *Categorical data.* Categorical data is similar in flavor to text data. However, a data object is not a document containing terms, but rather a vector of characteristics, each of which takes on non-numerical labels. A particular example is the Congressional voting dataset [UCI]. Each data object is a Congressman, and the vector of characteristics is how he voted on every bill or law put through Congress. The true clustering of Congressmen is their political party affiliations. We show that our spectral algorithm can also be applied in this scenario. Again, only one cut is necessary as it defines a 2-clustering of the data. In Table VI, we see that we do better than both COOLCAT [BLC02] and ROCK [GRS00].

Spectral	COOLCAT '02	ROCK '00
0.480	0.498	0.499

Table VI. Congressional Voting Data (Entropy)

We also applied our algorithm to the Mushroom data set [UCI], which consists of 8,124 mushrooms, each described by 22 features – such as odor (which takes on values such as `almond`, `anise`, `creosote`) and habitat (which takes on values such as `grasses`, `leaves`, or `meadows`). We represented each mushroom as a vector and each possible value as a coordinate. Thus, each mushroom is described by a binary vector. Each mushroom is labeled either poisonous or edible; we consider this the true clustering of the data. The COOLCAT and ROCK algorithms have been applied to this dataset by [ATMS04], who also introduce LIMBO, a categorical clustering algorithm, and apply it to this dataset. In Table VII, we show the results of the experiment. The precision and recall were measured; we perform better than ROCK and COOLCAT in both measures, but LIMBO outperforms us in both measures.

4.2.3 *Gene expression data.* A microarray chip is a solid surface upon which spots of DNA are attached in a matrix-like configuration. By exposing the chip to RNA, the expression level (roughly the activity of the gene) can be determined

	Spectral	COOLCAT '02	ROCK '00	LIMBO '04
Precision	0.81	0.76	0.77	0.91
Recall	0.81	0.73	0.57	0.89

Table VII. Mushroom Data

for each gene on the microarray chip. A seminal paper [GST+99] proposed an approach to discovering new subtypes of cancer by clustering microarray data. The approach is to cluster gene expression data from several patients with a certain type of cancer. If a strong clustering exists, the clustering might designate different subtypes of cancer. This relies on the the hypothesis that gene expression levels can distinguish different subtypes of cancer.

They tested the validity of this approach by applying it to a known sub-classification of leukemia. Two distinct subtypes of leukemia are: acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML). Golub et al. asked: could the approach to cancer classification correctly find the two known subtypes of ALL and AML? To this end, they prepared microarray chips for 38 bone marrow samples. Each chip contained roughly 7,000 human genes. Each bone marrow sample came from either one of 27 ALL patients or 11 AML patients. The gene expression data thus can be described by a 38 by 7,000 matrix M. Golub et al. clustered the 38 row vectors of this matrix using self-organizing maps [TSM+99]. The confusion matrix of their clustering is shown in Table VIII.

	ALL	AML
C_1	26	1
C_2	1	10

Table VIII. Confusion matrix for Golub et al. clustering

The clustering (C_1, C_2) almost exactly obeys the ALL/AML distinction. Golub et al. also provide a list of genes that are highly expressed in C_1, but not in C_2, and vice versa – they posit that the expression level of these genes distinguish between AML and ALL.

We ran the spectral algorithm on the gene expression data [Gol], which we pre-processed as follows. First, the data was normalized: the expression level of each gene was normalized over the 38 samples such that the mean was zero and the standard deviation was one.[6] Then, we created a 38 by 14,000 matrix N; the jth column in the original gene expression matrix, $M(\cdot, j)$ corresponds to the two columns $N(\cdot, 2j-1)$ and $N(\cdot, 2j)$. The two columns in N separate the negative and positive values in M: if $M(i,j) < 0$, then $N(i, 2j-1) = |M(i,j)|$ and if $M(i,j) > 0$, then $N(i, 2j) = M(i,j)$. The similarity between two samples (i.e. two rows of N) is just the inner product. Thus, if two samples have a large inner product, they have a similar gene expression profile. The confusion matrix for the first cut found by the spectral algorithm is shown in Table IX.

[6]Golub et al. do the same normalization

	ALL	AML
C_3	18	1
C_4	9	10

Table IX. Confusion matrix for Spectral 2-clustering

While C_3 is almost a pure ALL cluster, the clustering (C_3, C_4) does not obey the ALL/AML class boundary. Interestingly, recursing on the cluster C_4 gives a 3-clustering that does obey the ALL/AML class boundary. Table X shows the confusion matrix for this 3-clustering.

	ALL	AML
C_3	18	1
C_5	0	10
C_6	9	0

Table X. Confusion matrix for Spectral 3-clustering

A random 3-clustering does not obey the class boundary, so clearly the 3-clustering found by the spectral algorithm obeys the natural properties of the data. But why does the 2-clustering found by the spectral algorithm not respect this distinction? Preliminary investigations suggest that the 2-clustering finds distinguishing genes that are more statistically significant than the Golub clustering; it seems worthwhile to fully investigate the biological significance of this finding.

4.3 Merge phase in practice

The experiments in Section 4.2 imply that a good clustering exists in the tree created by the spectral algorithm. When the number of desired clusters k is small (i.e. in the SMART dataset or the 20 newsgroups dataset), finding a good k-clustering in the tree is not difficult. The only 2-clustering in a hierarchical clustering tree is the first partition, and there are only two 3-clusterings to examine. However, when the number of desired clusters is high, there are an exponential number of possible clusterings.

The Boley webpage dataset and Reuters dataset provide a good test for the merge phase, since the number of true clusters in these datasets is high. We show that for these datasets, the objective functions find a good flat clustering in the tree.

4.3.0.1 *Web pages:.* Recall that the Boley dataset consists of 185 webpages, each of which fall into 10 distinct classes. Section 4.2 showed that a good 16-clustering of the dataset exists in the tree.

We ran the same experiments J1-J11 on this dataset, but after constructing the tree via the spectral algorithm in the divide phase, we applied dynamic programs for three different objective functions (k-means, min-sum and min-diameter[7]) in the

[7]We did not apply the correlation clustering objective function, because we cannot control the number of clusters it finds. Any comparison to an objective function with a fixed number of

merge phase. We set k, the number of clusters desired, to 10. We also determined the 10-clustering in the tree with the lowest entropy. The results appear in Table XI. We see that k-means and min-sum generally perform better than min-diameter. What is most interesting is that the clustering obtained by either min-sum or k-means is only slightly worse than the best clustering in the tree. Indeed, in 7 of 11 experiments, the clustering obtained by min-sum or k-means was exactly the best clustering in the tree. Table XII shows the confusion matrix from the clustering obtained in experiment J4. All classes except for C_9 and C_{10} have a clear corresponding cluster $\hat{C}_i$. The weight on the diagonal is 131, meaning that $131/185 \approx 71\%$ of the articles are correctly classified.

	J1	**J2**	**J3**	**J4**	**J5**	**J6**	**J7**	**J8**	**J9**	**J10**	**J11**
k-means	1.00	0.93	0.88	0.81	1.00	0.93	0.84	0.83	0.95	0.71	1.07
min-sum	0.98	0.93	0.88	0.78	0.98	0.92	0.84	0.83	0.94	0.71	1.10
min-diam	1.04	1.10	0.96	1.04	1.10	1.00	1.05	1.23	1.24	0.83	1.16
best in tree	0.98	0.93	0.88	0.78	0.96	0.91	0.84	0.83	0.92	0.71	1.05

Table XI. Webpage dataset: Objective function performance in the merge phase (Entropy)

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
$\hat{C}_1$	**18**	0	3	0	1	0	1	0	0	0
$\hat{C}_2$	0	**17**	0	0	0	2	0	0	0	0
$\hat{C}_3$	0	1	**13**	2	0	0	0	0	0	1
$\hat{C}_4$	0	0	0	**10**	0	0	0	1	7	1
$\hat{C}_5$	0	0	0	0	**18**	0	1	0	0	0
$\hat{C}_6$	0	1	0	0	0	**13**	1	0	0	0
$\hat{C}_7$	0	0	1	3	0	1	**15**	0	0	0
$\hat{C}_8$	0	0	1	0	0	0	0	**12**	0	0
$\hat{C}_9$	1	0	0	2	1	0	1	4	**3**	3
$\hat{C}_{10}$	0	0	1	2	0	0	0	2	8	**12**

Table XII. Webpage dataset: Confusion matrix for min-sum clustering on experiment J4

4.3.0.2 *Reuters:.* The Reuters dataset also contains a large number of classes – 8,654 of the 21,578 articles are uniquely assigned to 65 labels. We chose all 1,832 articles that were assigned to the following 10 labels: coffee, sugar, trade, ship, money-supply, crude, interest, money-fx, gold, or gnp. We formed term vectors by removing stopwords and stemming words. We also removed all words that occur in less than 2% of the articles or more than 50% of the articles. A complete hierarchical tree was computed in the divide phase using the spectral algorithm, and we used dynamic programs in the merge phase to compute flat clusterings for the k-means, min-sum, and min-diameter objective functions, for $k = 10, 15$ and 20. The results appear in Table XIII. We do not perform as well as in the webpage

clusters would be unfair.

dataset. However, the entropy of the clusterings found by k-means and min-sum are not too far from the entropy of the best clustering in the tree.

We give the confusion matrix for the k-means clustering for $k = 10$ in Table XIV; only $\hat{C}_4$, $\hat{C}_5$, $\hat{C}_7$, and $\hat{C}_{10}$ do not clearly correspond to any class. The weight along the diagonal is 1068, meaning that $1068/1832 \approx 58\%$ of the articles are correctly classified.

	$k = 10$	$k = 15$	$k = 20$
k-means	1.02	0.92	0.84
min-sum	1.05	0.90	0.79
min-diam	1.10	0.94	0.80
best in tree	0.99	0.84	0.76

Table XIII. Reuters dataset: Objective function performance in the merge phase (Entropy)

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
$\hat{C}_1$	**96**	4	3	19	0	0	0	0	0	1
$\hat{C}_2$	0	**109**	0	9	0	0	0	0	0	0
$\hat{C}_3$	4	1	**203**	29	0	4	1	8	0	2
$\hat{C}_4$	0	6	2	**71**	0	100	0	2	0	0
$\hat{C}_5$	2	2	59	3	**75**	13	11	11	2	15
$\hat{C}_6$	0	0	0	3	0	**198**	0	0	0	0
$\hat{C}_7$	1	0	55	5	7	14	**160**	148	0	49
$\hat{C}_8$	0	0	0	1	1	1	0	**60**	0	0
$\hat{C}_9$	3	4	1	5	0	5	0	1	**90**	0
$\hat{C}_{10}$	8	9	10	11	14	20	39	29	7	**6**

Table XIV. Reuters dataset: Confusion matrix for k-means experiment with $k = 10$

5. CONCLUSION

We have presented a divide-and-merge methodology for clustering, and shown an efficient and effective spectral algorithm for the divide phase. For the merge phase, we have described dynamic programming formulations that compute the optimal tree-respecting clustering for standard objective functions. A thorough experimental evaluation of the methodology shows that the technique is effective on real-world data.

This work raises several additional questions: are the tree-respecting clusterings (for a suitable choice of objective function) provably good approximations to the optimal clusterings? Does the tree produced by the spectral algorithm contain a provably good clustering? Which objective functions are more effective at finding the true clustering in the tree in practice?

REFERENCES

M. Anderberg. *Cluster Analysis for Applications.* Academic Press, 1973.

Periklis Andritsos, Panayiotis Tsaparas, Rene J. Miller, and Kenneth C. Sevcik. Limbo: Scalable clustering of categorical data. In *International Conference on Extending Database Technology*, 2004.

N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science*, pages 238–247, 2002.

F. Beil, M. Ester, and X. Xu. Frequent term-based text clustering. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 436–442, 2002.

D. Barbara, Y. Li, and J. Couto. Coolcat: an entropy-based algorithm for categorical clustering. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 582–589, 2002.

D. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 626–635, 1997.

M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 524–533, 2003.

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 *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329, 1992.

D. Cheng, R. Kannan, S. Vempala, and G. Wang. Eigencluster. `http://eigencluster.csail.mit.edu`.

I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, 2001.

E.D. Demaine and N. Immorlica. Correlation clustering with partial information. In *Proceedings of the 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 1–13, 2003.

W.F. de la Vega, M. Karpinski, C. Kenyon, and Y. Rabani. Approximation schemes for clustering problems. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 50–58, 2003.

D. Emanuel and A. Fiat. Correlation clustering–minimizing disagreements on arbitrary weighted graphs. In *Proceedings of the 11th European Symposium on Algorithms*, pages 208–220, 2003.

G. H. Golub and C. F. Loan. *Matrix Computations.* Johns Hopkins, third edition, 1996.

T. R. Golub. Golub leukemia data.

Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.

T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

T. Hofmann. The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data. In *International Joint Conference on Artificial Intelligence*, pages 682–687, 1999.

J.A. Hartigan and M.A. Wong. A k-means clustering algorithm. In *Applied Statistics*, pages 100–108, 1979.

A. Jain and R. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31, 1999.

A. Kumar, S. Sen, and Y. Sabharwal. A simple linear time $(1+\epsilon)$-approximation algorithm for k-means clustering in any dimensions. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 454–462, 2004.

R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad, and spectral. *Journal of the ACM*, 51(3):497–515, 2004.

B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22, 1999.

K. Lang. 20 newsgroups data set. `http://www.ai.mit.edu/people/jrennie/20Newsgroups/`.

D. Lewis. Reuters data set. `http://www.daviddlewis.com/resources/testcollections/reuters21578/`.

A. Nickerson, N. Japkowicz, and E. Milios. Using unsupervised learning to guide re-sampling in imbalanced data sets. In *Proceedings of the Eighth International Workshop on AI and Statitsics*, pages 261–265, 2001.

G. Salton. SMART Data Set. `ftp://ftp.cs.cornell.edu/pub/smart`.

S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–566, 1976.

C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 1948.

A. Sinclair and M. Jerrum. Approximate counting, uniform generation, and rapidly mixing markov chains. *Information and Computation*, 82:93–133, 1989.

M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. In *Proceedings of the 23d Annual International ACM Conference on Research and Development in Information Retrieval*, pages 208–215, 2000.

C. Swamy. Correlation clustering: Maximizing agreements via semidefinite programming. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pages 519–520, 2004.

J. Theiler and G. Gisler. A contiguity-enhanced k-means clustering algorithm for unsupervised multispectral image segmentation. In *Proceedings of the Society of Optical Engineering*, pages 108–111, 1997.

P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. Lander, and T. Golub. Interpreting patterns of gene expression with self-organizing maps; methods and application to hematopoietic differentiation. *Proc. Nat. Acad. Sci*, 96:2907–2912, 1999.

UCI. UCI Machine Learning Repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

C. J. Van Rijsbergen. *Information Retrieval, 2nd edition.* Dept. of Computer Science, University of Glasgow, 1979.

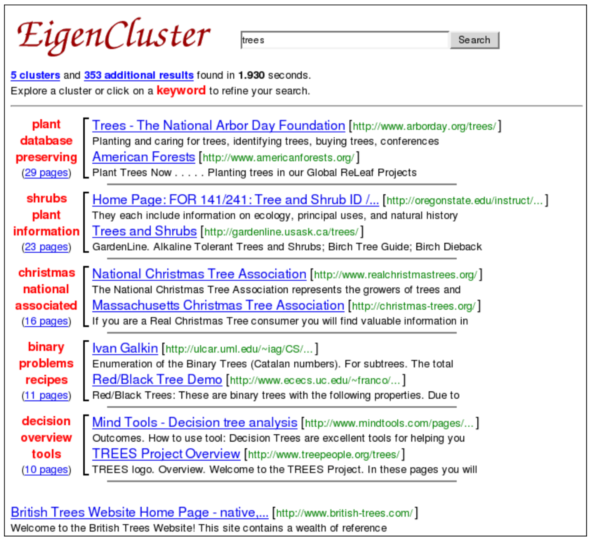
W. Wong and A. Fu. Incremental document clustering for web page classification. In *IEEE International Conference on Information Society in the 21st Century: Emerging Technologies and New Challenges*, 2000.

H. Zha, C. Ding, M. Gu, X. He, and H. Simon. Spectral relaxation for k-means clustering. In *Neural Information Processing Systems*, pages 1057–1064, 2001.

O. Zamir, O. Etzioni, O. Madani, and R. M. Karp. Fast and intuitive clustering of web documents. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 287–290, 1997.

Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 515–524, 2002.

6. EIGENCLUSTER EXAMPLE SEARCHES



EigenCluster

trees Search

5 clusters and 353 additional results found in 1.930 seconds.
Explore a cluster or click on a keyword to refine your search.

plant database preserving (29 pages)
Trees - The National Arbor Day Foundation [http://www.arborday.org/trees/]
Planting and caring for trees, identifying trees, buying trees, conferences
American Forests [http://www.americanforests.org/]
Plant Trees Now . . . . . Planting trees in our Global ReLeaf Projects

shrubs plant information (23 pages)
Home Page: FOR 141/241: Tree and Shrub ID /... [http://oregonstate.edu/instruct/...]
They each include information on ecology, principal uses, and natural history
Trees and Shrubs [http://gardenline.usask.ca/trees/]
GardenLine. Alkaline Tolerant Trees and Shrubs; Birch Tree Guide; Birch Dieback

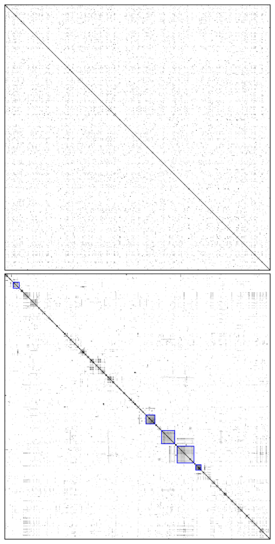
christmas national associated (16 pages)
National Christmas Tree Association [http://www.realchristmastrees.org/]
The National Christmas Tree Association represents the growers of trees and
Massachusetts Christmas Tree Association [http://christmas-trees.org/]
If you are a Real Christmas Tree consumer you will find valuable information in

binary problems recipes (11 pages)
Ivan Galkin [http://ulcar.uml.edu/~iag/CS/...]
Enumeration of the Binary Trees (Catalan numbers). For subtrees. The total
Red/Black Tree Demo [http://www.ececs.uc.edu/~franco/...]
Red/Black Trees: These are binary trees with the following properties. Due to

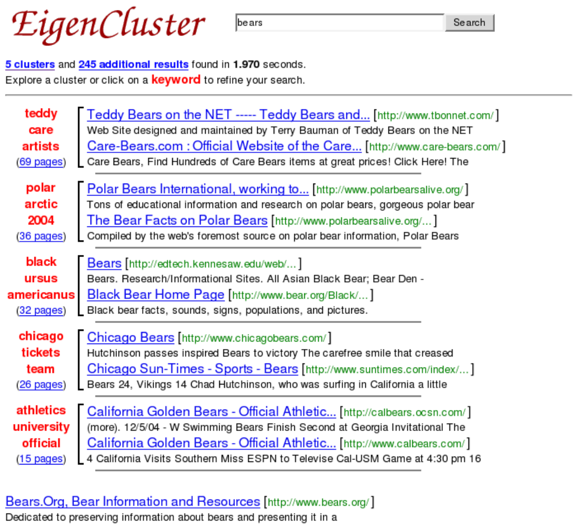
decision overview tools (10 pages)
Mind Tools - Decision tree analysis [http://www.mindtools.com/pages/...]
Outcomes. How to use tool: Decision Trees are excellent tools for helping you
TREES Project Overview [http://www.treepeople.org/trees/]
TREES logo. Overview. Welcome to the TREES Project. In these pages you will

British Trees Website Home Page - native,... [http://www.british-trees.com/]
Welcome to the British Trees Website! This site contains a wealth of reference

(a) Query: trees



(b) Before/after: trees



EigenCluster

bears Search

5 clusters and 245 additional results found in 1.970 seconds.
Explore a cluster or click on a keyword to refine your search.

teddy care artists (69 pages)
Teddy Bears on the NET ----- Teddy Bears and... [http://www.tbonnet.com/]
Web Site designed and maintained by Terry Bauman of Teddy Bears on the NET
Care-Bears.com : Official Website of the Care... [http://www.care-bears.com/]
Care Bears, Find Hundreds of Care Bears items at great prices! Click Here! The

polar arctic 2004 (36 pages)
Polar Bears International, working to... [http://www.polarbearsalive.org/]
Tons of educational information and research on polar bears, gorgeous polar bear
The Bear Facts on Polar Bears [http://www.polarbearsalive.org/...]
Compiled by the web's foremost source on polar bear information, Polar Bears

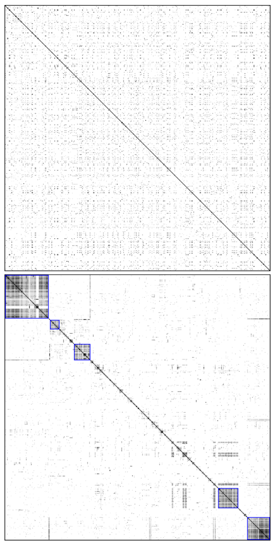
black ursus americanus (32 pages)
Bears [http://edtech.kennesaw.edu/web/...]
Bears. Research/Informational Sites. All Asian Black Bear; Bear Den -
Black Bear Home Page [http://www.bear.org/Black/...]
Black bear facts, sounds, signs, populations, and pictures.

chicago tickets team (26 pages)
Chicago Bears [http://www.chicagobears.com/]
Hutchinson passes inspired Bears to victory The carefree smile that created
Chicago Sun-Times - Sports - Bears [http://www.suntimes.com/index/...]
Bears 24, Vikings 14 Chad Hutchinson, who was surfing in California a little

athletics university official (15 pages)
California Golden Bears - Official Athletic... [http://calbears.ocsn.com/]
(more). 12/5/04 - W Swimming Bears Finish Second at Georgia Invitational The
California Golden Bears - Official Athletic... [http://www.calbears.com/]
4 California Visits Southern Miss ESPN to Televise Cal-USM Game at 4:30 pm 16

Bears.Org, Bear Information and Resources [http://www.bears.org/]
Dedicated to preserving information about bears and presenting it in a

(c) Query: bears



(d) Before/after: bears

Fig. 5. Example EigenCluster searches