A Multi-level Approach for Document Clustering

Suely Oliveira and Sang-Cheol Seok

The University of Iowa, Iowa City IA 52242, USA

Abstract. The divisive MinMaxCut algorithm of Ding et al. [3] produces more accurate clustering results than existing document cluster methods. Multilevel algorithms [4, 1, 5, 7] have been used to boost the speed of graph partitioning algorithms. We combine these two algorithms to construct faster and more accurate algorithm. In this new algorithm, the original graph is coarsened, partitioned by the divisive MinMaxCut algorithm and then decoarsened. A refining algorithm is also applied to improve the accuracy at each level.

1 Introduction

Clustering is the task of classifying a collection of objects, such as documents, into natural categories. Diagnostic tasks in medicine involve classifying a set of symptoms according to the cause and treatment methods. Data mining frequently involves separating documents into different categories so as to provide only relevant information for a user's query. We do not expect that classification or clustering algorithms will ever be 100% accurate, and the related optimization problems are frequently NP-hard. However, we do expect to find algorithms which are highly accurate that are nevertheless very efficient for practical clustering problems.

A variety of algorithms have been proposed and are in widespread use, including the K-means method and its variants [9], the Ratio Cut method [4], and the Normalized Cut method [12]. A recent method that has been proposed is based on the computation of eigenvectors which is called the MinMaxCut algorithm proposed by Ding et al. [3].

The MinMaxCut algorithm proposed by Ding et al. [3] outperformed the existing document clustering methods in accuracy. The algorithm leads to better accuracies because it aims to satisfy both the following desirable properties of a clustering of objects: (P1) nodes in the same cluster are similar, and (P2) nodes in the different clusters are dissimilar. The K-means method mainly achieves (P1), while RatioCut and NormalizedCut mainly achieve (P2). Two-way MinMaxCut is a clustering method which splits the whole cluster into two smaller clusters while divisive MinMaxCut is a K-way MinMaxCut which has as many as K clusters by splitting one of the current clusters repeatedly. A different version of K-way MinMaxCut may have a certain stopping criterion to decide how many clusters it has. The two key algorithms in this divisive method are how to select a

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3514, pp. 204–211, 2005.

cluster to split and how to split the cluster. Two-way MinMaxCut is used to split one cluster into two clusters for divisive MinMaxCut algorithm. Ding et al. [3] presents an efficient way to select one cluster using the sum of similarities of all pairs of each cluster. The two-way MinMaxCut algorithm they use is closely related to the spectral graph partitioning algorithm, which is finding the *Fiedler vector* [11], the second smallest eigenvector of the Graph Laplacian associated with a graph. Finding this eigenvector is, however, expensive and spends the majority of the time required by spectral partitioning techniques.

A class of graph partitioning algorithms [4, 1, 5, 7] coarsen the graph by collapsing vertices and edges, partition the smaller graph, and then decoarsening the partition on the coarsened graph. These methods are called multilevel graph partitioning schemes. These speed up the execution time. Later papers [1] use refining steps to achieve more accurate partitions Hendrickson and Leland [5] presented an improved algorithm for coarsening step by using edge and vertex weights to capture the collapsing of the vertex and edges. The coarsening algorithm of Karypis and Kumar [7] has much smaller coarsened graphs than other multilevel algorithms. They also present a variant of Kernigan-Lin (KL) refinement method which is faster than the original KL algorithm.

In this paper we present a divisive MinMaxCut algorithm to which a multilevel scheme, that is, coarsening, clustering and decoarsening, is applied. Our algorithm take advantage of both fast multilevel algorithm and accurate divisive MinMaxCut algorithm. This method uses edge weights to match 80% to 90% of the nodes by using edges with the highest weights. The other nodes with smaller edge weights, are grouped randomly. The coarsest graphs have less than one hundred nodes. The K-way MinMaxCut method produces a fairly good cut for this coarsest graph. Then during the decoarsening step, the cut for the coarsest graph is decoarsened and refined for all level-ups. The initial cut from the coarsest graph is getting more accurate as the level goes up and finally becomes a very accurate cut for the original graph. The major advantage of multilevel scheme is speed. Furthermore the cut which is decoarsened and refined at last for the original graph is more accurate than that of Ding et al's version of divisive MinMaxCut algorithm.

Our algorithm was tested on newsgroup articles in 20 newsgroups. The accuracy is mostly increasing first and then decreasing and the speed slowly improves as the number of levels increases. With more than 2 levels our algorithm takes less than half the time of Ding et al.'s algorithm. Like other multilevel algorithms, constructing the coarsened graphs takes most of the execution time. So for further improvement, a more efficient coarsening method is highly desirable.

2 *K*-Way MinMaxCut and Related Issues

In this section we first of all introduce the divisive MinMaxCut algorithm. Then we discuss some issues related to Multilevel approach.

2.1 Two-Way and *K*-Way MinMaxCut

Divisive MinMaxCut algorithm is based on the two-way MinMaxCut. It starts with a similarity matrix $S = (s_{ij})$, where $s_{ij} \geq 0$ stands for the similarity between node i and j. The more two nodes are similar the bigger s_{ij} is. Similarly, the less two nodes are related the smaller s_{ij} is. It aims to maximize the sum of similarities in a cluster and to minimize the sum of similarities of two nodes in different clusters. We define the sum of similarities between two clusters A and B as $s(A,B) = \sum_{i\in A, j\in B} s_{ij}$, and the sum of similarities of a cluster A as $s(A,A) = \sum_{i\in A, j\in A} s_{ij}$. The two-way MinMaxCut aims to minimize the objective function

$$J_{MMC} = \frac{s(A,B)}{s(A,A)} + \frac{s(A,B)}{s(B,B)} = \frac{s(A,\bar{A})}{s(A,A)} + \frac{s(B,\bar{B})}{s(B,B)} \tag{1}$$

Note that there are many objective functions that satisfy both (P1) and (P2). However, a solution to a continuous relaxation of J_{MMC} can be computed efficiently [2, 3]. An indicator vector q is used as the clustering solution. Each component of q gets one of two discrete numbers ($q_i = a$ if $i \in A$, otherwise $q_i = b$). Instead, if we relax the condition that $q_i = a$ or b to $q_i \in \mathbf{R}$, it is well known [3] that the optimal solution of (1) is the eigenvector q_2 associated with the second smallest eigenvalue of the system below with $D = diag(d_1, d_2, \cdots, d_n)$ and $d_i = \sum_j s_{ij}$:

$$(D - S)q = \lambda Dq. \tag{2}$$

After searching an optimal dividing point i_{cut}, the final cut is computed

$$A = \{i \mid q_2(i) \leq q_2(i_{cut})\}, \qquad B = \{i \mid q_2(i) > q_2(i_{cut})\}. \tag{3}$$

The optimal dividing point i_{cut} is the minimizer of the objective function. The dividing point can be computed in $O(N^2)$ time with a linear search. Note that, however, this does not guarantee that the cut after this linear search has no room to be more accurate. We talk about the refinement method in section 2.3.

A good generalization of the two-way MinMaxCut objective function (1) is:

$$J_{MMC}(C_1, ..., C_K) = \sum_{k=1}^{K} \frac{s(C_k, \bar{C}_k)}{s(C_k, C_k)} = \sum_{1\leq p\leq q} J_{MMC}(C_p, C_q). \tag{4}$$

where $\bar{C}_i = \bigcup_{j\neq i} C_j$.

2.2 Divisive MinMaxCut

Divisive MinMaxCut algorithm repeatedly performs two main steps. One is selecting a cluster to split and the other is applying the two-way MinMaxCut algorithm. We discussed the two-way MinMaxCut algorithm which is how to split one cluster into two clusters in section 2.1. So we now talk about the way to select the next cluster to split. Ding et al. [3] suggests 5 plans:

Size-priority cluster split: Choose the biggest current cluster.

Average similarity: Select the cluster with smallest $\bar{s}_{kk} := s(C_k, C_k)/|C_k|^2$.

Cluster cohesion: Select the cluster which has the smallest cohesion.

Similarity-cohesion: Select the cluster with the smallest $\bar{s}_{kk} \times$ cohesion.

Greedy: Select the cluster which leads to the minimum objective function value.

The best results are obtained by average similarity cluster selection [3]. For the stopping criterion, we use a user-selected number K; that is, the algorithm selects and splits clusters until there are K clusters. Another criterion is using a threshold on the objective function value which increases monotonically as the number of leaf clusters increase. For details, see [3].

2.3 Refinement

The Kernighan-Lin (KL) refinement [8] method was successfully applied for refining partitions of graphs in a multilevel algorithm [7]. We use the KL algorithm for our refining scheme. KL starts with an initial partition. It iteratively searches for nodes from each cluster of the graph if swapping of a node to one of the other $K-1$ clusters leads to a better partition. For each node, there would be more than one cluster to give smaller objective function value than the current cut. So the node moves to the cluster that give the biggest improvement. The computation for each node takes only $O(N)$ complexity. So the overall complexity per round does not exceed $O(N^2)$. The iteration terminates when it does not find any node to improve the partition or it finds the predefined number of nodes which lead to a better result. We may apply several iterations of KL to find a better partition.

2.4 Multi-level Approach: Coarsening and Decoarsening

The basic concept is that when we have a big graph $G_0 = (V_0, E_0)$ to cluster, then we construct a smaller graph $G_1 = (V_1, E_1)$ each of whose vertices is a group of several vertices from G_0. We can apply a clustering method to this smaller graph, and transfer this partition to the original graph. This idea is very useful because smaller matrices requires much less time. The process we construct the smaller matrix is called coarsening, and the reverse process is called decoarsening. We can recursively coarsen $G_i = (V_i, E_i)$ to get $G_{i+1} = (V_{i+1}, E_{i+1})$.

The decoarsening step is the way back to the original graph by going through the graphs $G_i, G_{i-1}, \cdots, G_0$. Note that even if the cluster in G_i is a local optimum, the cluster in the next finer level G_{i-1} might not be a local optimum. So we need to check if there is any room to improve the partition in this finer level. Refinement methods are used as the level goes up by one. For more details in the context of graph partitioning, refer to [6, 7].

3 Specific Algorithm and Computational Experiments

3.1 Description

The algorithm consists of three main steps. (1) Coarsening; (2) Clustering the coarsest graph into K subgraphs with the divisive method; (3) Decoarsening and refinement.

The Coarsening and Decoarsening steps are implemented by multiplying special matrices $E_1, \cdots, E_{level}$. We select two nodes to collapse by checking edges from the highest edge weight. Then one column of E is filled with two 1's for the two nodes and 0's for the rest. We collapse less than 90% of nodes by checking the edge weights. The rest of nodes which are not grouped are collapsed randomly. We finally construct $S_0, S_1, \cdots, S_l$ and $E_1, \cdots, E_l$ such as $S_i = E_i' * S_{i-1} * E_i$ and $i = 1, \cdots, l$. The coarsest similarity matrix is used to get the initial partition Cut. During the Decoarsening step the Cut in the current level is multiplied by the proper E for the partition in the next finer level.

We use Divisive MinMaxCut algorithm with average similarity selection scheme. In this algorithm we don't use any specific stopping criterion but predefined number of clusters, let us say K. That is the divisive method stops when we have K clusters in it.

3.2 Source of Data and Preprocessing

The experiments is performed on newsgroup articles in 20 newsgroups (datasets available online [10]). We focus on two sets of 5-clusters cases. The choice of $K = 4, 8$ where the clustering results are less sensitive to cluster section is avoided. The newsgroups chosen are listed in Table 1.

Clusters in M5 overlap at medium level. Meanwhile, clusters in L5 overlap at large. From each set of the newsgroups, we construct two datasets of different sizes: (B) randomly select 100 articles from each newsgroup. (U) randomly select 200, 140, 120, 100, 60 articles from each of the 5 newsgroups, respectively. Dataset(B) has clusters of equal sizes, which is presumably easier to cluster. Dataset(U) has clusters of significantly varying sizes, which is presumably difficult to cluster. Therefore we have 4 newsgroup- cluster size combination categories.

After documents from each category are extracted, we construct a word-document matrix $W = (x_1, \cdots, x_N)$ using standard **tf.idf** scheme. After each

Table 1. 10 newsgroups at different overlapping levels

Dataset M5	Dataset L5
NG2: comp.graphics	NG2: comp.graphics
NG9: rec.motorcycles	NG3: comp.os.ms-windows
NG10: rec.sport.baseball	NG8: rec.autos
NG15: sci.spaces	NG13: sci.electronics
NG18: talk.politics.mideast	NG19: talk.politics.misc

document of W is normalized to 1 using L_2 norm, document-document similarities are calculated as $S = W^T W$.

3.3 Result and Analysis: Accuracy, Time, Objective Function

This section has two main parts. One is comparison of two methods, Divisive MinMaxCut and Multilevel divisive MinMaxCut. The other is some important issues regarding the multi level approach. We constructed 2 different randomly sampled datasets from each category. The average of the time and accuracy are compared with the result from [3]. Their results come from the average of 5 different datasets from each category. For both cases the selection algorithm for the next cluster to split is average similarity selection.

We compare accuracies, time consuming and saturation for Divisive MinMaxCut and Multilevel divisive MinMaxCut in Table2. I/F stands for 'initial' and 'final' accuracies. D and MD mean divisive MinMaxCut and Multilevel divisive MinMaxCut respectively. Initial accuracy of D is the accuracy for the clustering generated by the eigenvector without any refinement. Initial accuracy of MD is the accuracy measured just after clustering the coarsest graph and decoarsening the partition without refinement. The number of levels used for MD is 4. We will see the results with different levels after this. All time consumings are in second and all accuracies are in percent.

Accuracy. We see the initial accuracy of D is mostly better than that of MD but the final accuracy of MD is mostly better than that of M. The reason is that the initial partition of D comes from the whole graph while the initial partition of MD comes from much smaller graph, the coarsest graph. Instead, MD is refining at each level up of decoarsening step.The final result is much different and improved from the initial partition.

Time. MD finishes clustering much faster. MD spends most time for coarsening. For example, it takes 1.63 seconds to go through the first two steps for a dataset of M5B and coarsening takes 1.62 of 1.63 and partitioning takes the rest, 0.01. And 0.19 second is spent for decoarsening and refinement. Note that refinement step for D takes various time consuming depending on the refining scheme and the number of rounds it has.

Saturation comparison: Improvement of accuracy does not exceed some point even though refining step is applied repeatedly. This upper bound is called the

Table 2. Comparison of plain divisive(D) and multidivisive(MD) methods for different categories. Time and accuracy are measured for with/without refinement(I/F) cases. Upper bound of accuracy, saturation(sat), for both cases D and MD are experimented

	D Accuracy I/F	MD Accuracy I/F	D time I/F	MD time I/F	D sat	MD sat
M5B	83.5/91.7	77.2/98.4	2.91/28.53	1.63/1.82	92.5	99.2
M5U	69.3/72.4	70.5/87.5	3.83/55.48	2.99/3.17	91.7	97.42
L5B	88.4/91.7	62.6/81.4	2.01/28.82	1.60/1.71	81.4	95.0
L5U	74.8/74.1	58.5/83.2	3.72/54.94	3.04/3.18	79.0	93.39

saturation of objective function. Saturation of D comes from [3] and that of MD comes from the best result among all levels of each category. This shows MD provides more accurate result than D.

Now we focus on MD itself. We will see how different results MD gives us depending on the various levels. The main focuses here are (a) the relation between time and number of levels and (b) accuracy and the number of levels.

(a) time and level: Table 3 shows how much time is spent for each of three steps. We used two datasets. One is balanced and the other is unbalanced. Time for each of all three steps is measured. As we see the first step, coarsening, takes the most time. Partitioning step takes less time as the number of levels increases. When the number of nodes in the level is less than 100 it takes at most one hundredth second. In both cases the total time consumed decreases slowly as the number of levels increases.

Table 3. Time consuming of all three steps (coarsening/partitioning/decoarsening) of MD for balanced(B) and unbalanced(UB) cases with various levels

level	1	2	3	4
B	1.43/0.33/0.19	1.59/0.05/0.18	1.58/0.00/0.18	1.57/0.01/0.19
UB	2.65/0.53/0.22	2.98/0.08/0.27	3.02/0.01/0.38	2.96/0.00/0.18

(b) accuracy and level: As you see in Table 4, the accuracy and the number of levels curve kind of upside down parabola on average. All 8 dataset are listed, where each 2 datasets are from one of 4 categories. The best accuracy comes from level 3 or 4 where the number of nodes in the coarsest graph is around 50.

We conclude that Multilevel divisive MinMaxCut works very well when compared to the existing cut-based document clustering methods in time and accuracy. More research may be necessary on the relationship between the number of levels and accuracy; that is, how we can decide the optimal number of levels depending the size of the original graph.

Table 4. Relationship between accuracy and level for MD

level	1	2	3	4	5	6
M5Ba	80.4	98.2	98.2	99.2	96.2	68.2
M5Bb	91.8	94.8	95.8	97.6	92.6	60.2
L5Ba	93.4	80.5	95.0	71.2	89.2	82.0
L5Bb	85.6	88.8	54.6	91.6	62.8	65.4
M5Ua	76.0	86.8	79.4	97.3	80.6	91.26
M5Ub	94.3	76.6	97.4	77.7	88.2	78.9
L5Ua	75.5	69.4	82.9	78.9	66.1	54.8
L5Ub	72.3	64.7	93.4	87.6	72.3	59.4

Acknowledgments

We would like to acknowledge the support of the National Science Foundation for this work through grant DMS-02-13305 and the help from Dr. David Eichmann for generating the similarity matrices.

References

1. T. Bui and C. Jones. A heuristic for reducing fill in sparse matrix factorization. In *6th SIAM Conf. Parallel Processing for Scientific Computing*, pages 445–452, 1993.
2. C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A min-max cut algorithm for graph partitioning data clustering. In *ICDM 2001, Proceedings IEEE International Conference on Data Mining, 2001*, pages 107–114. IEEE, 2001.
3. C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A minmaxcut spectral method for data clustering and graph partitioning. Technical Report 54111, LBNL, December 2003.
4. L. Hagen and A. Kahng. Fast spectral methods for ratio cut partitioning and clustering. In *Proceedings of IEEE International Conference on Computer Aided Design*, pages 10–13. IEEE, 1991.
5. B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. Technical Report SAND93-1301, Sandia National Laboratories, 1993.
6. M. Holzrichter and S. Oliveira. A graph based davidson algorithm for the graph partitioning problem. *International Journal of Foundations of Computer Science*, 10:225–246, 1999.
7. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report 95-035, Department of Computer Science, University of Minnesota, Minneapolis, MN, 1998.
8. B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 1970.
9. J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proc. Fifth Berkeley Symp. Math. Statistics and Probability*, 1:281–296, 1967.
10. A. McCallum. A toolkit for statistical language modeling, text retrieval, classification and clustering., 1996. Available on WWW at URL `http://www.cs.cmu.edu/~mccallum/bow`.
11. A. Pothen, H. D. Simon, and Kang-Pu K. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, 1990.
12. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug 2000.