Hierarchical Taxonomy Preparation for Text Categorization Using Consistent Bipartite Spectral Graph Copartitioning

Bin Gao, Tie-Yan Liu, *Member*, *IEEE Computer Society*, Guang Feng, Tao Qin,
Qian-Sheng Cheng, and Wei-Ying Ma, *Member*, *IEEE Computer Society*

Abstract—Multiclass classification has been investigated for many years in the literature. Recently, the scales of real-world multiclass classification applications have become larger and larger. For example, there are hundreds of thousands of categories employed in the Open Directory Project (ODP) and the Yahoo! directory. In such cases, the scalability of classification methods turns out to be a major concern. To tackle this problem, hierarchical classification is proposed and widely adopted to get better trade-off between effectiveness and efficiency. Unfortunately, many data sets are not explicitly organized in hierarchical forms and, therefore, hierarchical classification cannot be used directly. In this paper, we propose a novel algorithm to automatically mine a hierarchical structure from the flat taxonomy of a data corpus as a preparation for the adoption of hierarchical classification. In particular, we first compute matrices to represent the relations among categories, documents, and terms. And, then, we cocluster the three substances at different scales through consistent bipartite spectral graph copartitioning, which is formulated as a generalized singular value decomposition problem. At last, a hierarchical taxonomy is constructed from the category clusters. Our experiments showed that the proposed algorithm could discover very reasonable taxonomy hierarchy and help improve the classification accuracy.

Index Terms—Clustering, data mining, singular value decomposition, text processing.

1 Introduction

Multiclass classification has been actively investigated for many years. On one hand, several m-way classifiers such as k-nearest neighbors (k-NN) [5] and Naive Bayes (NB) [5], [18] were developed. On the other hand, people have developed some strategies to extend well-performed binary classifiers such as Support Vector Machines (SVM) [21] to the multiclass case. To the best of our knowledge, the earliest and the most popular strategy is one-against-rest, in which each of the binary classifiers is trained to distinguish one category from all other categories and all these classifiers will make YES/NO decisions on a given instance in the test phase. As a result, the instance will be classified into the category corresponding to the highest confidence score. Many benchmark evaluations [15], [21] have shown that one-again-rest SVM classifiers (denoted by flat SVM) lead to higher classification accuracy as compared to NB and k-NN.

The flat SVM works pretty well when the category number is small. However, in recent years, the problem scale of multiclass classification has been larger and larger. For instance, the well-known large-scale Web directories Open Directory Project (http://dmoz.org/) and Yahoo! directory (http://dir.yahoo.com/) have 118,488 and 292,216 categories, respectively. In such large-scale cases, flat SVM will suffer from its poor scalability because its training complexity is $O(MN)$ and its test complexity is $O(M)$, where M and N are the total numbers of categories and documents in the corpus [16], [24].

To tackle this problem, people proposed using the inner hierarchical structures among the categories to divide the classification task. This resulted in the so-called hierarchical SVM, in which a SVM classifier is trained to distinguish each child category only from other categories with the same parent, rather than from all other categories in the corpus. For the test phase, *pachinko-machine* search is used (i.e., a lower-level SVM classifier is activated only if its parent gives a YES decision). Empirical studies on large-scale data sets have shown that hierarchical SVM often excels flat SVM in both complexity and accuracy [6], [16]: On one hand, local feature selection and parameter tuning might improve the effectiveness; on the other hand, the training complexity is reduced because the training documents of each local task are much less than the flat setting, and the *pachinko-machine* search will significantly reduce the time consumption for testing.

However, as we all know, it is not a necessity that data corpora have explicitly-given hierarchical taxonomies. Therefore, for many cases, hierarchical classification can not be adopted even if we know it is superior to flat classification. To tackle this problem, one possible solution is to mine a hierarchical configuration by our own and use it to organize the hierarchical classifiers. In other words, we need to preprocess the data corpus so as to do some essential preparations for hierarchical classification.

- *B. Gao and Q.-S. Cheng are with LMAM, Department of Information Science, School of Mathematical Sciences, Peking University, Haidian District, Beijing, 100871, P.R. China.*
E-mail: gaobin@math.pku.edu.cn, qcheng@pku.edu.cn.
- *T.-Y. Liu and W.-Y. Ma are with Microsoft Research Asia 5F, Sigma Center, No. 49, Zhichun Road, Haidian District, Beijing, 100080, P.R. China. E-mail: {t-tyliu, wyma}@microsoft.com.*
- *G. Feng and T. Qin are with MSP Laboratory, Department of Electronic Engineering, Tsinghua University, Haidian District, Beijing, 100084, P.R. China. E-mail: {fengg03, qinshitao99}@mails.tsinghua.edu.cn.*

Manuscript received 20 Nov. 2004; revised 28 Mar. 2005; accepted 1 Apr. 2005; published online 19 July 2005.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDESI-0440-1104.

1041-4347/05/$20.00 © 2005 IEEE Published by the IEEE Computer Society

To our knowledge, there have been some attempts in this direction, although not many. Vural and Dy [22] proposed a method named divided-by-two (DB2), where the categories of a data corpus is recursively divided into two subsets until each subset consists of only one category. As a result, a binary taxonomy tree is generated. The subset division is done through clustering categories based on the similarities among their documents. Godbole et al. [8], [9] proposed another method to automatically generate the hierarchical taxonomy. They used an NB classifier to quickly compute a category-by-category confusion matrix and adopted a graph-based clustering method to extract a hierarchical taxonomy.

From the above previous works, we can see that clustering might be a very important step for hierarchical taxonomy mining. However, the clustering methods used in these works are not very effective because they have not made full use of the information contained in categories, documents, and terms. This is very similar to what happened in the field of document clustering. As we know, most early document clustering algorithms directly defined the similarity of documents by the similarity of their term representations (i.e., the cosine distance between two TF-IDF [1] vectors) until the concept of document-term coclustering was proposed [2], [3], [25]. In the philosophy of coclustering, the similarity between documents is defined by their term representations while the similarity between terms is defined by their occurrences in documents. In other words, document similarity and term similarity are defined in a reinforcing manner. In such a way, it has been shown that the clustering performance can be greatly improved [2], [3], [25]. Similarly, we believe that, by using the information contained in categories, documents, and terms for category clustering rather than only using the category information or document information, we can get more reasonable results on taxonomy mining. Actually, in our previous work [23], we have proposed a coclustering framework named ReCoM with the similar philosophy. However, in that work, heterogeneous objects were treated as homogeneous objects under some heuristic scaling factors. Sometimes this assumption might be too strong and the tuning of the scaling factor is theoretically difficult. Therefore, we want to avoid it in the method proposed in this paper.

In particular, in this paper, we use one bipartite graph to represent the relationship between categories and documents and use another to represent the relationship between documents and terms. Then, we partition these two bipartite graphs consistently by solving a generalized singular value decomposition problem. Experiments showed that our method could not only lead to a very reasonable taxonomy hierarchy, but also result in higher classification accuracy than previous methods.

The rest of this paper is organized as follows: In Section 2, the background knowledge and related work are introduced. Then, Section 3 describes the proposed method in detail, while experimental results are discussed in Section 4. Some concluding remarks and future work directions are listed in the last section.

2 Related Works

As the proposed method is illumined by the thought of document-term coclustering, in this section, we will introduce some essential background knowledge and review a representative work in this direction.

For the first step, we need to introduce the widely-used document representation in information retrieval and text categorization, vector space model (VSM) [1], because most the previous works on coclustering were built on top of it. The main idea of VSM is to directly treat terms in the documents as features, so as to map a document to a vector in the feature space. Specifically, in VSM, $D = d_1, d_2, \ldots, d_n$ denotes the documents in the corpus and $T = w_1, w_2, \ldots, w_t$ denotes the terms. Then, each document d_i in D can be represented by a t-dimensional vector $d_i = x_{i1}, x_{i2}, \ldots, x_{it}$, where x_{ij} is the term frequency [1] (or TF-IDF [1]) of term w_j in document d_i.

The problem of document-term coclustering is to cluster the documents $D = d_1, d_2, \ldots, d_n$ and terms $T = w_1, w_2, \ldots, w_t$ in a reinforcing manner. There are several methods to tackle this problem [2], [3], [13], [25], among which Dhillon et al. proposed an approach based on bipartite spectral graph partitioning. To illustrate how their algorithm works, we need to introduce some essential knowledge on graph partitioning as below.

A graph $G = (V, E)$ is composed by a set of vertices $V = \{1, 2, \ldots, |V|\}$ and a set of edges $E = \{< i, j > | i, j \in V\}$, where $|V|$ represents the number of vertices. If using E_{ij} to denote the weight of edge $< i, j >$, we can further define the adjacency matrix M of the graph by

$$M_{ij} = \begin{cases} E_{ij} & if \ < i, j > \in E \\ 0, & otherwise. \end{cases} \tag{1}$$

Suppose the vertex set V is partitioned into two subsets V_1 and V_2, then the corresponding *cut* is defined as:

$$cut(V_1, V_2) = \sum\nolimits_{i \in V_1, j \in V_2} M_{ij}. \tag{2}$$

The above definition can be easily extended to k subsets:

$$cut(V_1, V_2, \ldots, V_k) = \sum\nolimits_{\eta < \theta} cut(V_\eta, V_\theta). \tag{3}$$

If the vertices of a graph can be decomposed into two disjoint subsets such that no two vertices within the same set are adjacent, the graph is named a bipartite graph. In other words, a bipartite is a triplet $G = (V_1, V_2, E)$, where V_1 and V_2 are two vertex sets, within each of which no vertices are adjacent and E is a set of edges connecting vertices from different vertex sets, i.e., $E = \{< i, j > | i \in V_1, j \in V_2\}$. Specifically, in [2], an undirected bipartite graph, as shown in Fig. 1, is used to represent the relationship between documents and terms. Here, V_1 and V_2 are replaced with D and T, which represent the subsets of document vertices and term vertices, respectively, and E denotes the set of edges $\{< d_i, w_j > | d_i \in D, w_j \in T\}$. An edge $< d_i, w_j >$ exists if and only if the term w_j occurs in document d_i and its weight E_{ij} equals the corresponding term frequency.

If we use B to denote the document-by-term matrix in which B_{ij} equals the edge weight E_{ij}, the adjacency matrix of the bipartite graph, as shown in Fig. 1, can be written as:

$$M = \begin{array}{cc} & \begin{array}{cc} D & T \end{array} \\ \begin{array}{c} D \\ T \end{array} & \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix} \end{array}, \tag{4}$$

where the vertices have been ordered such that the first n vertices index the documents (denoted by D) while the last t index the terms (denoted by T).

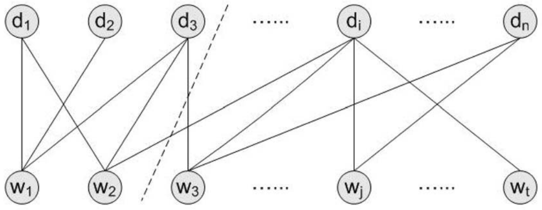


Fig. 1. The document-term bipartite graph.

The key idea of [2] is to find a partition of the vertices in the graph such that the *cut* as defined in (2) (or normalized cut [19], ratio cut [11], min-max cut [4], etc.) can be minimized. For instance, if the dashed line in Fig. 1 shows the very partition that minimizes the *cut*, we will obtain two subsets, $\{d_1, d_2, d_3, w_1, w_2\}$ and $\{d_4, \ldots, d_n, w_3, \ldots, w_t\}$. Therefore, the documents are clustered into two groups, $\{d_1, d_2, d_3\}$ and $\{d_4, \ldots, d_n\}$, while the terms are clustered into $\{w_1, w_2\}$ and $\{w_3, \ldots, w_t\}$ at the same time. It was proved in [2], [10], [19] that the eigenvector associated with the second smallest eigenvalue λ_2 of the generalized eigenvalue problem $L\omega = \lambda Q\omega$ is an optimal embedding for the partition of the vertices that minimizes the *cut*. Here, Q is a diagonal matrix with $Q_{ii} = \sum_k E_{ik}, L = Q - M$ is the Laplacian matrix, and ω is a column vector. After some trivial deduction, the above problem can be converted to a singular value decomposition (SVD) problem, which can be computed more efficiently. For the details of this algorithm, please refer to [2], [25]. Although there are some other works on document-term coclustering, since their basic philosophy is not relevant to our proposed method, we will not review them in details in this section. One can find these reference works in [3], [13].

3 Coclustering-Based Hierarchical Taxonomy Mining

As shown in the introduction and the related works, clustering might be a key technology in unsupervised taxonomy mining. In this section, we will also use this methodology to mine hierarchical taxonomy from data corpora. Specifically, we propose a novel clustering algorithm, which coclusters categories, documents, and terms based on consistent bipartite spectral graph copartitioning. We will first explain how this new algorithm is formulated from Section 3.1 to 3.3, and then discuss how to construct the taxonomy hierarchy based on the output of this algorithm in Section 3.4.

Before going into the details of the algorithm description, we need to give some additional notations. Besides the document $(D = \{d_1, d_2, \ldots, d_n\})$ and term $(T = \{w_1, w_2, \ldots, w_t\})$ representations as mentioned in Section 2, for our application, text categorization, we also have one more substance, category. Each document will be assigned a category label from the set $C = \{c_1, c_2, \ldots, c_m\}$, where m is the number of categories. The coclustering will be conducted on all these three substances of C, D, and T.

3.1 Coclustering-Based on Category-Document Bipartite Spectral Graph Partitioning

It is not difficult to acknowledge the analogy between document-category coclustering and document-term coclustering: Categories are similar because the documents in them are similar, while documents are similar because they are likely to be classified into similar categories. Therefore, we can construct a bipartite graph to represent the relationship between categories and documents as well (see Fig. 2).

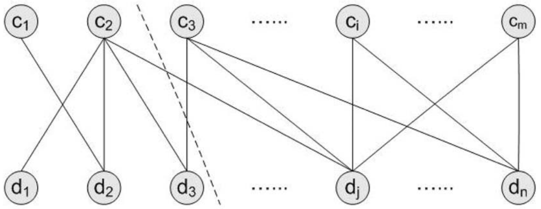


Fig. 2. The category-document bipartite graph.

The category-by-document matrix A can be easily built according to the information from the corpus. In this matrix, rows correspond to categories and columns to documents. Each element A_{ij} indicates the correlation between document d_j and category c_i. If document d_j belongs to k categories $c_1, c_2, \ldots, c_k$, the weights $A_{1j}, A_{2j}, \ldots, A_{kj}$ are set to $1/k$, and the other elements of the jth column of matrix A are set to zero. Then we can write the adjacency matrix of the bipartite graph in Fig. 2 as follows:

$$M = \begin{array}{cc} & \begin{array}{cc} C & D \end{array} \\ \begin{array}{c} C \\ D \end{array} & \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \end{array}, \tag{5}$$

where the vertices have been ordered such that the first m vertices index the categories (denoted by C) while the last n index the documents (denoted by D).

It is easy to understand that the coclustering of categories and documents can also be mapped to solving the generalized eigenvalue problem $L\omega = \lambda Q\omega$. Here, Q is a diagonal matrix with $Q_{ii} = \Sigma_k M_{ik}$ and $L = Q - M$ is the Laplacian matrix. Similar to Dhillon's method [2] described in Section 2, we can solve this problem by computing a singular value decomposition. The corresponding algorithm is given as below:

Algorithm 1.

1. Given A, form its normalized version,

$$\hat{A} = P^{-1/2} A R^{-1/2},$$

where P and R are diagonal matrices with $P_{ii} = \Sigma_j A_{ij}, R_{ii} = \Sigma_j A_{ji}$.

2. Compute the left singular vector u_2 and the right singular vector v_2 corresponding to the second largest singular value of $\hat{A}$ and get the eigenvector corresponding to the second smallest eigenvalue of $L\omega = \lambda Q\omega$ as $\omega_2 = [P^{-1/2}u_2 \;\; R^{-1/2}v_2]^T$.
3. Cluster on the one-dimensional data $P^{-1/2}u_2$ and $R^{-1/2}v_2$ to obtain the desired bipartition of categories and documents, respectively.

The same manner as described in [2] can be used to extend this algorithm to its multipartitioning version. While we use the second singular vector to partition the graph into two parts, we could use $l = \lceil \log_2 k \rceil$ singular vectors on each side ($u_2, u_3, \ldots, u_{l+1}$ and $v_2, v_3, \ldots, v_{l+1}$) when partitioning the graph into k parts. In such a way, Algorithm 1 can be updated as below:

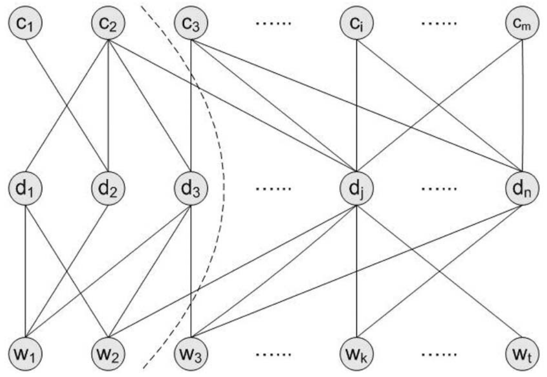


Fig. 3. The category-document-term tripartite graph.

Algorithm 2.

1. Given A, form P, R, and $\hat{A} = P^{-1/2}AR^{-1/2}$.
2. Compute $l = \lceil \log_2 k \rceil$ singular vectors of $\hat{A}, u_2, u_3, \ldots, u_{l+1}$ and $v_2, v_3, \ldots, v_{l+1}$, and form Ω as in (6):

$$\Omega = \begin{bmatrix} P^{-1/2}U \\ R^{-1/2}V \end{bmatrix}, \text{ with } \begin{cases} U = [u_2, u_3, \ldots, u_{l+1}] \\ V = [v_2, v_3, \ldots, v_{l+1}] \end{cases}. \quad (6)$$

3. Cluster on the l-dimensional data $P^{-1/2}U$ and $R^{-1/2}V$ to obtain the desired k-partitioning of categories and documents, respectively.

It seems that Algorithm 1 and 2 are natural and workable; however, we have to point out a critical problem of these formulations. Although, in the case of multilabel classification, the above methods might be able to cluster correlated categories together, in the single-label case, the graph is actually made up of m unconnected subgraphs corresponding to the m categories because the weights of those edges between a category and a document which does not belong to the category will be zero. Therefore, we cannot get a desirable partitioning because the connectivity of the graph is not guaranteed. To avoid this situation, one possible way is to further add the term information to the category-document graph so that the graph can be well connected.

3.2 Coclustering Based on Category-Document-Term Tripartite Spectral Graph Partitioning

As the similarity between documents can be defined not only by categories but also by terms, the bipartite graph proposed in Section 3.1 reflects only partial information of the data corpus. To compensate it and avoid the failure case in single-label categorization, a straightforward way is to leverage the term information so as to cocluster the category, document, and term at the same time.

Superimposing Fig. 2 upon Fig. 1, we will obtain a category-by-document-by-term tripartite graph as shown in Fig. 3. Here, a k-partite graph is a graph whose graph vertices can be partitioned into k disjoint sets so that no two vertices within the same set are adjacent.

It is easy to derive the adjacency matrix for the above graph:

$$M = \begin{array}{c} \\ C \\ D \\ T \end{array} \begin{array}{c} \begin{array}{ccc} C & D & T \end{array} \\ \begin{bmatrix} 0 & A & 0 \\ A^T & 0 & \alpha B \\ 0 & \alpha B^T & 0 \end{bmatrix} \end{array}, \quad (7)$$

where α is a weighting parameter and the vertices have been ordered such that the first m vertices index the categories (denoted by C), the next n index the documents (denoted by D), and the last t index the terms (denoted by T).

Up to now, by introducing the term information, we might have the chance to solve the failure case caused by lack of graph connectivity. Then, does everything go smoothly? And can the coclustering be worked out by solving the generalized eigenvalue problem corresponding to this adjacency matrix? Unfortunately, the answers to the above questions are once again negative. In fact, if we move the category vertices in Fig. 3 to the side of the term vertices, the original tripartite graph will turn out to be a bipartite graph. This can also be proved through the following deductions: In the tripartite case, as shown in Fig. 3,

$$Q = \begin{bmatrix} P & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & S \end{bmatrix} \text{ and } L = \begin{bmatrix} P & -A & 0 \\ -A^T & R & -\alpha B \\ 0 & -\alpha B^T & S \end{bmatrix}, \quad (8)$$

where P, R, and S are diagonal matrices such that $P_{ii} = \Sigma_j A_{ij}$, $R_{ii} = \Sigma_j A_{ji} + \alpha \Sigma_j B_{ij}$, and $S_{ii} = \alpha \Sigma_j B_{ji}$. Let vector $\omega = (x, y, z)^T$, where x, y, and z are column vectors of m, n, and t dimensions, respectively. Then, $L\omega = \lambda Q \omega$ may be written as:

$$\begin{bmatrix} P & -A & 0 \\ -A^T & R & -\alpha B \\ 0 & -\alpha B^T & S \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \lambda \begin{bmatrix} P & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & S \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \quad (9)$$

Assuming that P, R, and S are all nonsingular, the above equations can be rewritten as:

$$\begin{cases} P^{-1/2}Ay & = (1-\lambda)P^{-1/2}x \\ R^{-1/2}A^T x + \alpha R^{-1/2}Bz & = (1-\lambda)R^{-1/2}y \\ \alpha S^{-1/2}B^T y & = (1-\lambda)S^{-1/2}z. \end{cases} \quad (10)$$

If we set $u = P^{1/2}x$, $v = R^{1/2}y$, and $s = S^{1/2}z$, we will get the following equations after a little algebraic simplification:

$$\begin{cases} P^{-1/2}AR^{-1/2}v & = (1-\lambda)u \\ R^{-1/2}A^T P^{-1/2}u + \alpha R^{-1/2}BS^{-1/2}s & = (1-\lambda)v \\ \alpha S^{-1/2}B^T R^{-1/2}v & = (1-\lambda)s. \end{cases} \quad (11)$$

Uniting the first and the last formulae in (11) together and considering that P, R, and S are all diagonal matrices, we will have:

$$\begin{cases} \begin{bmatrix} P^{-1/2}AR^{-1/2} \\ S^{-1/2}(\alpha B)^T R^{-1/2} \end{bmatrix} v = (1-\lambda) \begin{bmatrix} u \\ s \end{bmatrix} \\ \begin{bmatrix} P^{-1/2}AR^{-1/2} \\ S^{-1/2}(\alpha B)^T R^{-1/2} \end{bmatrix}^T \begin{bmatrix} u \\ s \end{bmatrix} = (1-\lambda)v. \end{cases} \quad (12)$$

Let $\mu = (u, s)^T$ and

$$F = \begin{bmatrix} P^{-1/2}AR^{-1/2} \\ S^{-1/2}(\alpha B)^T R^{-1/2} \end{bmatrix}.$$

Then, we have:

$$\begin{cases} Fv = (1-\lambda)\mu \\ F^T \mu = (1-\lambda)v. \end{cases} \quad (13)$$

Therefore, $(1-\lambda)$ is the singular value of F and the generalized eigenvalue problem $L\omega = \lambda Q\omega$ is converted to an SVD problem, where $\mu = (u, s)^T$ and v are the left and right singular vectors, respectively. Note that u and s, representing categories and terms, respectively, are both embedded into the left singular vectors of F. That is, Fig. 3 is actually equivalent to a bipartite graph and we have to distinguish (by the weighting parameter α) the loss of cutting a category-document edge from the loss of cutting a document-term edge since they contribute to the same loss function. However, it is nontrivial to choose a proper value for α: if it is too small, the influence from the matrix B will make little sense and this graph will degenerate to the graph in Fig. 2. On the contrary, if it is too large, we will risk the situation of assigning all category vertices into one subset.

To summarize the above discussions, analyzing the matrix M in (7) by traditional spectral clustering methods does not work as it is expected. To tackle this problem, we will propose a novel algorithm based on consistent bipartite graph copartitioning. In this algorithm, we no longer need to tune the weight to balance document-category and document-term edges. Instead, we use generalized SVD to fuse the two bipartite graphs as shown in Fig. 1 and Fig. 2. The corresponding details will be described in the next section.

3.3 Coclustering Based on Consistent Bipartite Spectral Graph Copartitioning

As aforementioned, the motivation of coclustering categories, documents, and terms is to leverage the complementary information contained in Fig. 1 and Fig. 2. As we know, a document is the bridge between these two bipartite graphs. Therefore, if we can design a method to iteratively refine the partitioning of each graph by the information contained in the other, the stationary state will have the same partitioning of documents in these two graphs and the corresponding category clustering will be more reasonable and effective. We name this stationary state by consistent copartitioning of the two bipartite graphs.

In the following discussions, we will first show that standard SVD on Fig. 1 and Fig. 2 can hardly result in consistent copartitioning and, then, propose a new method to guarantee the consistent copartitioning.

3.3.1 Standard SVD Does Not Guarantee Consistent Copartitioning

To proceed with our discussions, we use M_1 to denote the adjacency matrix of Fig. 2 and M_2 to denote the adjacency matrix of Fig. 1:

$$M_1 = \begin{array}{cc} & \begin{array}{cc} C & D \end{array} \\ \begin{array}{c} C \\ D \end{array} & \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \end{array}, \quad M_2 = \begin{array}{cc} & \begin{array}{cc} D & T \end{array} \\ \begin{array}{c} D \\ T \end{array} & \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix} \end{array}, \tag{14}$$

where A and B are defined as aforementioned and the normalized forms of them are:

$$\hat{A} = P_1^{-1/2} A R_1^{-1/2}, \quad \hat{B} = P_2^{-1/2} B R_2^{-1/2}. \tag{15}$$

Here, P_1, R_1, P_2, and R_2 are diagonal matrices as defined in Algorithm 1.

Due to the spectral graph theory [2], [25], if we conduct the singular value decompositions on $\hat{A}$ and $\hat{B}$ as

$$\hat{A} = U_A \Sigma_A V_A^T, \quad \hat{B} = U_B \Sigma_B V_B^T, \tag{16}$$

unitary matrix U_A will embed the clustering information of categories, V_A and U_B will embed the clustering information of documents, and V_B will be the embedding for term clustering.

Our consistent copartitioning will ask for the same partitioning of documents in both graphs. Therefore, if using SVD for clustering as above, we need to guarantee the clustering results of the corresponding columns of V_A and U_B to be the same. However, it is not difficult to understand that the above consistence cannot be satisfied in many cases due to the different properties of the two pregiven matrices $\hat{A}$ and $\hat{B}$ [10]. To illustrate this, let us see an example as follows. Suppose matrices A and B of a single-labeled problem are:

$$A = \begin{array}{cc} & \begin{array}{cccc} d_1 & d_2 & d_3 & d_4 \end{array} \\ \begin{array}{c} c_1 \\ c_2 \\ c_3 \end{array} & \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}, \quad B = \begin{array}{cc} & \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{array}{c} d_1 \\ d_2 \\ d_3 \\ d_4 \end{array} & \begin{bmatrix} 4 & 3 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 \\ 0 & 0 & 3 & 4 & 0 \\ 0 & 0 & 0 & 4 & 3 \end{bmatrix} \end{array}.$$

After forming $\hat{A}$ and $\hat{B}$ and conducting the singular value decompositions on them, we will get:

$$V_A = \begin{bmatrix} 0.70711 & 0 & 0 & -0.70711 \\ 0.70711 & 0 & 0 & 0.70711 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

$$U_B = \begin{bmatrix} 0.70711 & 0 & 0 & -0.70711 \\ 0.70711 & 0 & 0 & 0.70711 \\ 0 & 0.70711 & -0.70711 & 0 \\ 0 & 0.70711 & 0.70711 & 0 \end{bmatrix}.$$

Now, we can see that the document clustering results based on U_B is $\{d_1, d_2; d_3, d_4\}$ while the clustering results based on V_A is $\{d_1, d_2, d_4; d_3\}$. Obviously, these two results are not consistent at all.

To overcome this problem, we suggest that we should relax the unitary constraint on V_A and U_B to get a tradeoff. In other words, for both graphs, we might not use the optimal embedding for clustering provided that these two embeddings can be consistent. As will be shown below, generalized SVD is just such a tool that can meet our requirement. Due to the following theory, we can find a consistent copartitioning of documents in these two graphs based on generalized SVD:

Theorem 1. *If we have $\hat{A} \in R^{m \times n}$ and $\hat{B} \in R^{n \times t}$, $m \leq n \leq t$, then there exists unitary matrices $U \in R^{m \times m}$, $V \in R^{t \times t}$ and invertible matrix $X \in R^{n \times n}$ such that:*

$$\begin{cases} \hat{A} = UCX^T \\ \hat{B} = XSV^T, \end{cases} \tag{17}$$

where $C = diag(c_1, c_2, \ldots, c_m), c_i \geq 0$ and $S = diag(s_1, s_2, \ldots, s_n), s_i \geq 0$.

Proof. Please refer to [10]. □

Theorem 1 indicates that there exists a partitioning of documents (whose embedding is X) that can meet the constraints of both graphs. The corresponding partitioning of categories and terms are embedded in U and V, respectively. However, a follow-up question is whether these embeddings are good and how to refine these embeddings if they are not. This will also be a critical part in the success of our proposed idea of consistent spectral bipartite graph copartitioning.

As shown in Theorem 1, X is an invertible matrix. This condition is much more relaxed as compared to standard

SVD where the embedding of document (i.e., V_A) will be a unitary matrix. Therefore, the coclustering information contained in X is not as explicit as in V_A. While the second singular vector of V_A has been a good embedding for document clustering, the same embedding can only be achieved by considering many nonorthogonal columns in X. As a chain reaction, this will also affect the explicitness of the coclustering information embedded in U. Therefore, we could not use the second columns of U, X, and V directly but should think of a way to extract useful information from them. In the mathematical manner, it is equal to finding a proper linear transformation.

For this purpose, we multiply CX^T and XS to form a mixture matrix H and conduct SVD decomposition on H to get two unitary matrices U_H and V_H:

$$H = CX^T XS, \quad H = U_H \Sigma_H V_H^T. \tag{18}$$

Then, we use U_H and V_H as the linear transformations to refine U and V:

$$U^* = UU_H, \quad V^* = VV_H. \tag{19}$$

Eventually, U^* and V^* are regarded as the modified embeddings for coclustering.

To summarize, the consistent bipartite spectral graph copartitioning algorithm (CBSGC) is given as below:

Algorithm 3.

1. Given A and B, form P_1, P_2, R_1, R_2, and $\hat{A}, \hat{B}$.
2. Compute GSVD of $\hat{A}, \hat{B}$ to get U, X, V, C, and S.
3. Form $H = CX^T XS$ and compute SVD of it to get U_H, V_H.
4. Form $U^* = UU_H, V^* = VV_H$ and take the second column vectors of them, u_2 and v_2, to form the normalized embedding vector
$$\omega_2 = [P_1^{-1/2} u_2 \;\; R_2^{-1/2} v_2]^T.$$
5. Cluster on the one-dimensional data $P_1^{-1/2} u_2$ and $R_2^{-1/2} v_2$ to obtain the desired bipartition of categories and terms, respectively.

Similar to the method in Section 3.1, we can extend the above bipartitioning algorithms to adapt the k-partitioning case. We would like to use $l = \lceil \log_2 k \rceil$ vectors on each side ($u_2, u_3, \ldots, u_{l+1}$ and $v_2, v_3, \ldots, v_{l+1}$) to obtain desired k-partitioning. As the extension form is quite similar to Algorithm 2, we omit the details here.

The reasoning of our consistent copartitioning process can be explained as follows: Since we can hardly get a consistent document partitioning if we partition both graphs optimally (corresponding to SVD decomposition), for compromise, we look for nonoptimal partitioning for each graph which can be consistent with each other. Although the partitioning for each graph is not optimal, when considering the two graphs as a whole, the resulting partitioning may be more effective.

For the above algorithm description, considering that our motivation is to get category clusters, one may argue that another approach to tackle the single-label trap is to use the category-term bipartite graph directly. We agree with that, but want to point out it is nontrivial to build the category-term relationship. If we use some heuristics to get it, we may lose some unrecoverable information. Comparatively, our approach is to use the consistency on the document partitioning to bridge categories and terms. This is a more natural and lossless way, because category and term are actually connected by documents in the real world. And Algorithm 3 is just one of the realizations of our idea. That is, we are not necessarily restricted to matrix multiplication or similar linear operations. The algorithm space that we can explore is very rich, under the same idea of consistency.

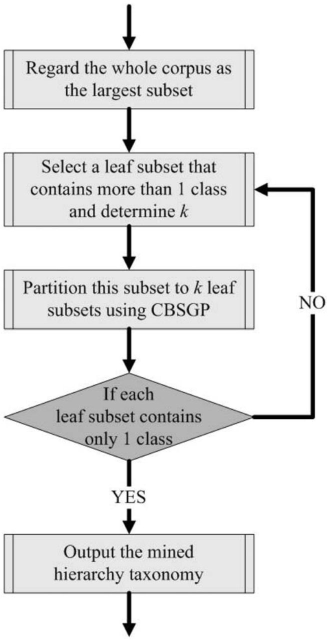

Regard the whole corpus as the largest subset
Select a leaf subset that contains more than 1 class and determine k
Partition this subset to k leaf subsets using CBSGP
If each leaf subset contains only 1 class
NO
YES
Output the mined hierarchy taxonomy

Fig. 4. The flow chart of building hierarchies.

3.4 Building Hierarchical Classifier

After designing the above consistent copartitioning algorithm, we can move onto the part of building the taxonomy hierarchy. As shown in Fig. 4, initially, we regard the whole collection of categories as the root of the hierarchical taxonomy and run the k-partitioning CBSGC algorithm on it to get several subsets of categories in level 2. This process is done recursively until each subset at the leaf nodes of the tree consists of only one category.

It is clear that k for different subsets should not be fixed. However, as we all know, it is a hard problem to determine k in clustering algorithms, although there have been some papers working on it [12]. In the field of spectral clustering, the eigengap [20] is often suggested as a way to determine the number of clusters; however, it does not always work, especially for real-world complex problems [17]. With the above concerns, we adopt a classical strategy to enumerate k in our method. We try different k values ($k = 1, 2, 3, \ldots, \mathrm{K}$, where K is the category number of the concerned subset) for clustering to get a *Je-k* curve, where Je is the minimum value for the objective function of a clustering algorithm. For example, Je for k-means [5] takes the form of:

$$Je = \min \sum_{i=1}^{k} \sum_{y \in \Gamma_i} \|y - \sigma_i\|^2, \tag{20}$$

where $\sigma_1, \sigma_2, \ldots, \sigma_k$ are centers of the clusters $\Gamma_1, \Gamma_2, \ldots, \Gamma_k$. Then, the very k at the inflexion point of the curve is chosen as the number of the clusters. For the above process, one may argue that the time complexity is high because of the enumeration. We agree with that, but want to point out that taxonomy mining could be an offline task. If we can get a reasonable taxonomy hierarchy and can thus improve

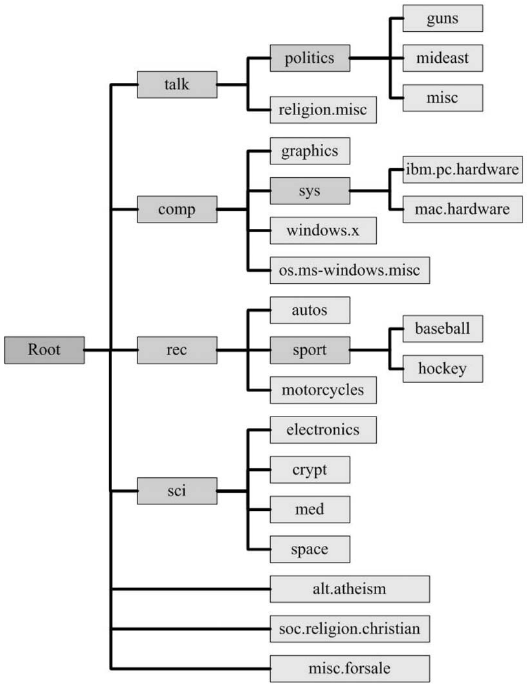


Fig. 5. The natural taxonomy of 20-newsgroups.

the classification efficiency and effectiveness by much, these additional offline data preparations are definitely worthy of being done.

After the hierarchical taxonomy is ready, we can build the hierarchical SVM classifiers. At each node of the mined taxonomy, a SVM classifier is trained to distinguish each of its child categories from other children of it. That is, the one-against-rest strategy is adopted at each node. The training process starts at the root node and covers all the categories using a breadth-first traversal of the taxonomy.

4 Experimental Results

In this section, we conducted experiments on two real data sets to show the outputs of taxonomy mining as well as the classification performance of hierarchical SVM on the mined taxonomy.

4.1 20-Newsgroups

The data set of 20-newsgroups contains about 20,000 articles evenly divided from 20 newsgroups. The human-labeled hierarchical taxonomy of 20-newsgroups is shown in Fig. 5. Our task is to mine this structure out by automatic methods.

As for the preparation, we randomly selected 100 articles per category to form the training set for taxonomy mining and classifier training. Then, we used the remaining 18,000 instances for testing. The experimental settings and external tools are summarized below:

1. For weighting, we used the method described in Section 3.1 to build the category-by-document matrix A and used term frequency to build the document-by-term matrix B. Here, we only used those terms that occur more than 10 times in the data set when generating matrix B.
2. We used the SVD and GSVD functions in the famous MATLAB (http://www.mathworks.com/) software to compute the corresponding matrix decompositions of the matrices.
3. At last, an accelerated k-means algorithm [7] was implemented to get the category clusters.
4. Microsoft Windows Server 2003 running on a workstation with 3.06GHz Xeon(TM) CPU and 3.87GB RAM was used for the experimental platform.

TABLE 1
A Mini Set Selected from 20-Newsgroups

Category Name	Notation
comp.sys.ibm.pc.hardware	C_4
comp.sys.mac.hardware	C_5
talk.politics.guns	C_{17}
talk.politics.mideast	C_{18}
talk.politics.misc	C_{19}

4.1.1 Coclustering Performance

In this section, we picked five categories (See Table 1) from 20-newsgroups to make up a mini set and randomly select 100 articles per category for coclustering. Here, we set $k = 2$ to test the coclustering performance of the four algorithms: Category-Document Bipartite spectral Graph Partitioning (CDBGP), Category-Document-Term Tripartite spectral Graph Partitioning (CDTTGP), Consistent Bipartite Spectral Graph Copartitioning (CBSGC), and Category-by-Category Confusion Matrix (CCCM) [8].

As the mini set is single-labeled, CDBGP made a random partition of the categories according to the arrangement of the documents. In our experiment, it regarded C_5 as a cluster and organized the others together. Then, we tuned different values of α to see the performance of the CDTTGP. From Table 2, we can see that the results are disappointing no matter how this parameter was set. In fact, when α is small, the algorithm degenerated to CDBGP. Otherwise, all categories were assigned to one cluster.

Comparatively, CBSGC and CCCM algorithms both exported a cluster of C_4, C_5 and another of C_{17}, C_{18}, C_{19}. The result was in accordance with our ground truth, which showed that CBSGC and CCCM outperformed the other two methods in category clustering.

4.1.2 The Mined Hierarchical Taxonomy

In this section, we ran the CBSGC algorithm on the whole data set of 20-newsgroups to mine the hierarchical taxonomy. During this process, we used the strategy described in Section 3.4 to determine the cluster number for each nonleaf node. For instance, Fig. 6 showed the *Je-k* curve for the root node, in which the number 4 at the inflexion point of the

TABLE 2
Performance under Different Values of α

α	Cluster 1	Cluster 2
1e-10	C_4,C_5,C_{18},C_{19}	C_{17}
0.01	C_4,C_5,C_{18},C_{19}	C_{17}
0.1	C_4,C_5,C_{17},C_{18},C_{19}	*null*
1	C_4,C_5,C_{17},C_{18},C_{19}	*null*
10	C_4,C_5,C_{17},C_{18},C_{19}	*null*
100	C_4,C_5,C_{17},C_{18},C_{19}	*null*
1e10	C_4,C_5,C_{17},C_{18},C_{19}	*null*

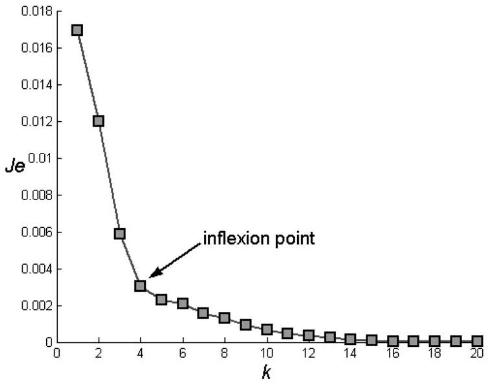


Fig. 6. The *Je-k* curve for the root node.

curve was chosen for k. To make the taxonomy compact and depress the computational cost, in our experiment, the recursion stopped when a subset contained no more than three categories. As a result, we mined the taxonomy hierarchy as shown in Fig. 7 for the 20-newsgroups data set.

We can see from this taxonomy that correlated categories such as *rec.**, *talk.**, *comp.**, and *sci.** were clustered together in different levels respectively except for some overlaps between *comp.** and *sci.**, and the single-handed categories such as *alt.atheism*, *misc.forsale*, and *soc.religion.christian* were separated solely from other clusters in certain levels. Further observation shows that low level categories such as *comp.sys.** and *rec.sport.** were also grouped together.

For comparison, we gave in Fig. 8 the taxonomy built in [8] by CCCM. We can see that Group 1 and Group 3 mix different kinds of categories together.

To summarize, the clusters of our mined taxonomy looked more reasonable, and the local properties of this hierarchy were more similar to those of the natural taxonomy in Fig. 5. The weakness of CBSGC is that the mined taxonomy often had more levels than the natural one.

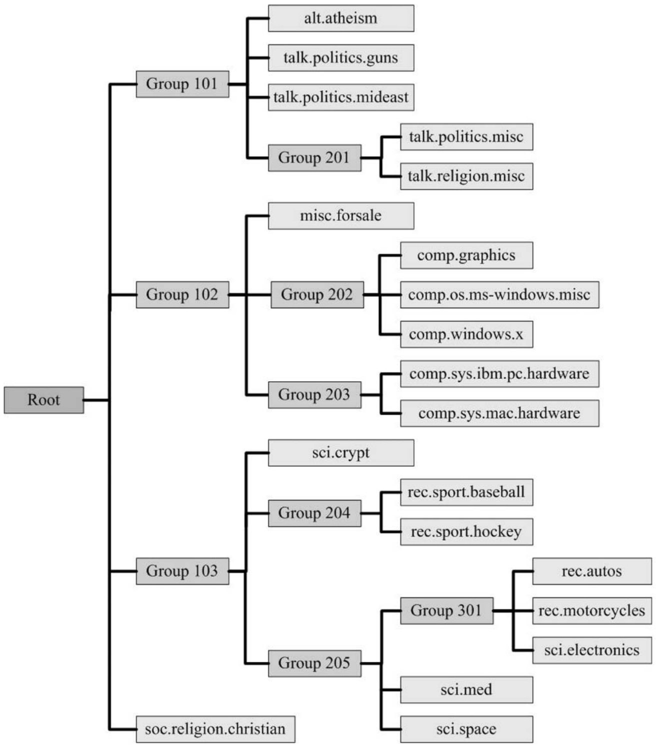


Fig. 7. The mined taxonomy of 20-newsgroups by CBSGC.

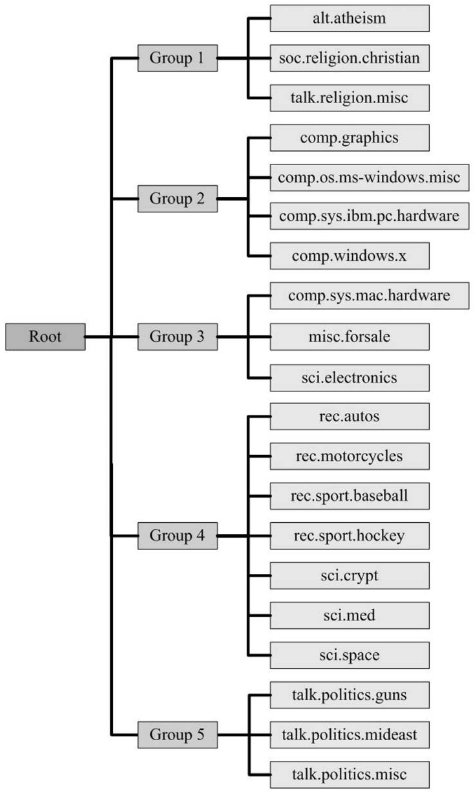


Fig. 8. The taxonomy of 20-newsgroups by CCCM.

4.1.3 Classification Performance

In this experiment, we investigated the accuracy of hierarchical classification with the mined taxonomy. As the baseline, we first constructed a flat SVM classifier. Then, we trained three hierarchical SVM classifiers according to the taxonomies shown in Fig. 8, Fig. 7, and Fig. 5, respectively. The corresponding classification results and time complexities of the above classifiers were summarized in Table 3, in which *Micro-F1* and *Macro-F1* are used for evaluating the classification performance [14].

As can be seen from Table 3, the performance of hierarchical classification based on CBSGC was better than both the flat classifier and the hierarchical classifier based on CCCM. Furthermore, we find that the classification performance of our method was very proximal to the natural hierarchical classifier. This showed in an indirect way that our taxonomy mining algorithm was very effective and the mined hierarchy was reasonable and meaningful. As one can see, the improvement brought by our method is not very significant (about 2 percent). The reason is that the 20-newsgroups corpus is somehow easy to classify. With a baseline of about 0.9 *F1* scores, the space of improvement could only be marginal.

For the time complexity of training and test, we could see that hierarchical classifiers outperformed flat classifiers significantly. Even if we further take the time of taxonomy mining (including the time spent on running generalized SVD and determining the cluster number k) into consideration, the overall complexity of CBSGC-based hierarchical SVM was still lower than flat SVM. It is very promising that

TABLE 3
Experimental Results of the Classifiers on 20-Newsgroups

Method	*Micro-F1*	*Macro-F1*	Training Time	Testing Time
Flat Classifier	0.902460	0.900139	2286.202s	61.777s
CCCM-based	0.906655	0.906656	413.467s	29.577s
CBSGC-based	0.919103	0.920266	551.546s	29.856s
Natural Hierarchy	0.922720	0.922692	523.074s	28.148s

(The time spent on mining the taxonomy by CBSGC was 273.902s.)

TABLE 4
The Selected Subtree of the Yahoo! Directory

1st level node	2nd level nodes	3rd level nodes
Science	*Mathematics*	*Algebra, Geometry*, etc. (Total 36 nodes.)
	Physics	*Electricity, Magnetism*, etc. (Total 29 nodes)
	Chemistry	*Biochemistry, Chemists*, etc. (Total 26 nodes.)

our proposed method can improve both the effectiveness and efficiency as compared to flat SVM classification.

4.2 A Subtree of the Yahoo! Directory

In this section, we conducted our algorithm on a subtree of the Yahoo! directory. It has been discovered [16] that the distribution of the Web pages (documents) in the Yahoo! directory is seriously skewed; that is, many leaf-node categories of the taxonomy contain very few documents while several common categories consist of thousands of documents. To avoid the influence of this ill distribution and show the validity of our algorithm, we selected the *Science* subtree for our experiment. The root node of the selected corpus was *Science*, and three children *Mathematics*, *Physics*, and *Chemistry* were selected as the second level nodes. Then, all the children of these three nodes were put in the third level and all other nodes were omitted. Table 4 gave a rough view of the selected subtree. (The data was crawled in Oct. 2004.) With such a setting, this subtree contained 91 leaf-node categories and 3,073 Web pages (documents). We randomly chose 80 percent (2,451) documents for training and the rest 20 percent (622) for testing. The weighting process for matrices A and B was the same as in Section 4.1.

4.2.1 Coclustering Performance and the Mined Hierarchical Taxonomy

After applying CBSGC on it, we got the embeddings of the category nodes in Fig. 9, where the second, third, and fourth columns of the embedding matrix U^* were plotted and the ground truth of category labels were shown (children of *Mathematics*, *Physics*, and *Chemistry* were denoted by “x,” “+,” and “.,” respectively).

Figs. 9a, 9b, 9c, and 9d showed the embeddings in different points of view. We could see that most of the nodes that belonged to the same category in level 2 were embedded very close to each other. For example, in Fig. 9d, most mathematical nodes were on the right side of the vertical axis; most physical nodes lay on the left of the vertical axis and below the horizontal axis; and most chemical nodes lay on the left of the vertical axis but over the horizontal axis. We could also see a few error-embedded nodes. For instance, the ”x” point near (0.2, 0) in Fig. 9d was *Wavelets*, a child of *Mathematics*, but it seemed more close to the children of *Physics* and *Chemistry*. In some sense, this might also be reasonable because from the bag-of-word view, ”wave” can be quite relevant to *Physics*.

Another finding is that there were some nodes far away from the majority, such as the ”x” point near (-0.4, 0.1) in Fig. 9d, which was *Mathematics/Education*. One explanation is that these categories had much more training documents than other categories. In other words, this charges upon the skewed category distribution. If we ran k-means on the embedding data as done in Section 4.1, these isolated points would tend to be clustered as separate clusters. To tackle this problem, we used the simplest method to get the clusters, that is, we chose the zero points of different dimensions as thresholds to partition the embedding points, and mined the taxonomy described in Table 5. From this table, we can see the clustering performance is quite good: Most categories are correctly clustered to their ground-truth labels, while only a few categories are wrongly clustered.

4.2.2 Classification Performance

In this section, we examine the classification performance of the mined taxonomy as shown in Table 5. We first constructed a flat SVM classifier composed of 91 one-again-rest SVM classifiers. Then, we trained two hierarchical SVM classifiers according to the taxonomies shown in Tables 4 and 5, respectively. The corresponding classification results and time complexities of the above classifiers were summarized in Table 6, from which we could see that the performance of hierarchical classification based on CBSGC was not only higher than flat SVM, but even better than the hierarchical classifier based on the natural taxonomy. This phenomenon is so promising for automatic taxonomy mining: some local structure of the artificial taxonomy might be unsuitable for machine classification. And for the complexity, we can get very similar conclusion to that from the 20-newsgroups data set: Even if we take the time for taxonomy mining into account, the time complexity of CBSGC-based hierarchical SVM classification is still much lower than flat SVM classification.

5 Conclusion and Future Work

In this paper, our target is to prepare a hierarchical taxonomy automatically in order to applying hierarchical classification. For this purpose, we proposed a method to cluster categories, documents, and terms by consistent

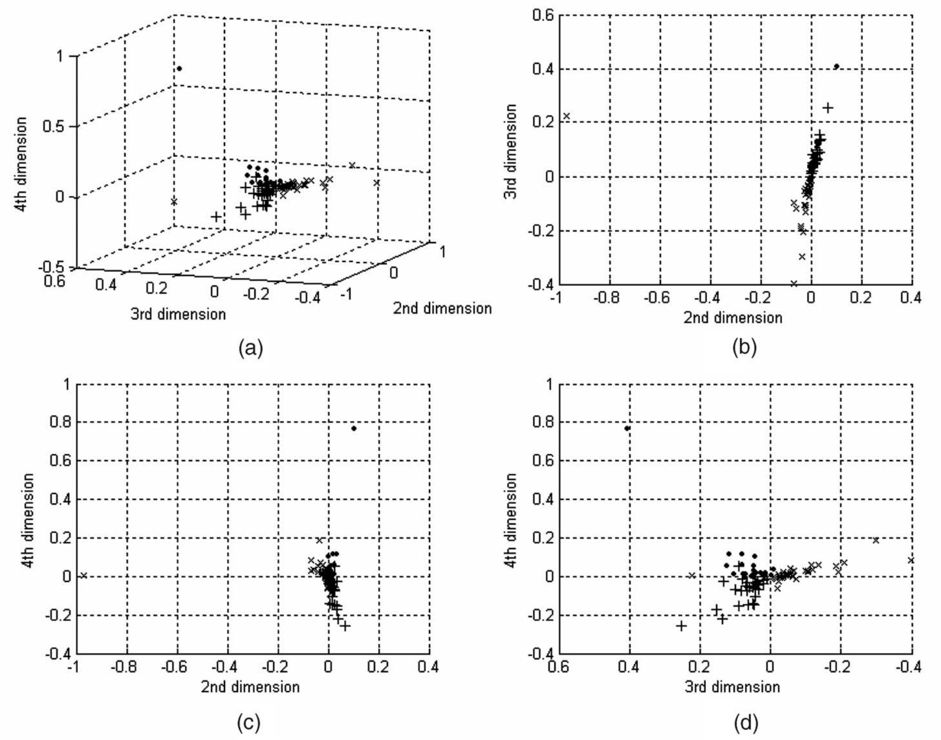


Fig. 9. The embeddings of the category nodes of the Yahoo! subtree.

TABLE 5
The Mined Subtree of the Yahoo! Directory

1^{st} level node	2^{nd} level nodes	3^{rd} level nodes
Science	*Mathematics*	*Algebra, Geometry*, etc. (37 nodes with 2 errors.)
	Physics	*Electricity, Magnetism*, etc. (30 nodes with 2 errors)
	Chemistry	*Biochemistry, Chemists*, etc. (24 nodes with 3 errors.)

TABLE 6
Experimental Results of the Classifiers on Subtree of the Yahoo! Directory

Method	*Micro-F1*	*Macro-F1*	Training Time	Testing Time
Flat Classifier	0.627668	0.456808	1005.997s	9.423s
CBSGC-based	0.664601	0.481123	651.606s	5.212s
Natural Hierarchy	0.661069	0.479177	669.724s	5.093s

(The time spent on mining the taxonomy by CBSGC was 104.517s.)

bipartite spectral graph copartitioning. Compared to previous approaches, this method used more information embedded in the data corpus to assist the taxonomy mining and the hierarchical classifier training. Experimental results showed that the proposed method worked well in category clustering and could produce a very similar hierarchical taxonomy to the natural hierarchy in both the appearance and the corresponding hierarchical classification performance. For future work, we plan to provide some reasonable objective functions for the consistent copartitioning problem and design fast algorithms to get the optimal solution. We would also like to find a more efficient way to select k automatically in the process of generating the hierarchical taxonomy.

Acknowledgments

This work was done when the first, the third, and the fourth authors were interns at Microsoft Research Asia. During the formulation of this paper, Hao WAN provided the authors with the code of the hierarchical SVM classifier and helped them understand and debug it.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval.* ACM Press, 1999.

[2] I.S. Dhillon, "Coclustering Documents and Words Using Bipartite Spectral Graph Partitioning," *Proc. SIGKDD '01,* 2001.

[3] I.S. Dhillon, S. Mallela, and D.S. Modha, "Information-Theoretic Co-Clustering," *Proc. SIGKDD '03,* pp. 89-98, 2003.

[4] C. Ding, X. He, H. Zha, M. Gu, and H. Simon, "A Min-Max Cut Algorithm for Graph Partitioning and Data Clustering," *Proc. IEEE Int'l Conf. Data Mining,* 2001.

[5] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification,* second ed. John Wiley & Sons Inc., 2001.

[6] S.T. Dumais and H. Chen, "Hierarchical Classification of Web Content," *Proc. SIGIR '00,* 2000.

[7] C. Elkan, "Using the Triangle Inequality to Accelerate k-Means," *Proc. Int'l Conf. Machine Learning,* 2003.

[8] S. Godbole, "Exploiting Confusion Matrices for Automatic Generation of Topic Hierarchies and Scaling Up Multi-Way Classifiers," technical report, IIT Bombay, 2002.

[9] S. Godbole, S. Sarawagi, and S. Chakrabarti, "Scaling Multi-Class Support Vector Machines Using Inter-Class Confusion," *Proc. SIGKDD '02,* 2002.

[10] G.H. Golub and C.F.V. Loan, *Matrix Computations,* third ed. Johns Hopkins Univ. Press, 1996.

[11] L. Hagen and A.B. Kahng, "New Spectral Methods for Ratio Cut Partitioning and Clustering," *IEEE Trans. Computer Aided Design,* vol. 11, pp. 1074-1085, 1992.

[12] G. Hamerly and C. Elkan, "Learning the *k* in *k*-Means," *Proc. Neural Information Processing Systems Conf.,* 2003.

[13] K. Kummamuru, A. Dhawale, and R. Krishnapuram, "Fuzzy Co-Clustering of Documents and Keywords," *Proc. IEEE Int'l Conf. Fuzzy Systems,* pp. 772-777, 2003.

[14] D.D. Lewis, "Evaluating Text Categorization," *Proc. Speech and Natural Language Workshop,* 1991.

[15] D.D Lewis, Y. Yang, T. Rose, and F. Li, "RCV1: A New Benchmark Collection for Text Classification Research," *J. Machine Learning Research,* vol. 5, pp. 361-397, 2004.

[16] T. Liu, Y. Yang, H. Wan, Q. Zhou, B. Gao, H. Zeng, Z. Chen, and W. Ma, "An Experimental Study on Large-Scale Web Categorization," *Proc. Int'l World Wide Web Conf.,* 2005.

[17] M. Meila and L. Xu, "Multiway Cuts and Spectral Clustering," http://www.stat.washington.edu/mmp/, 2004.

[18] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Computing Surveys (CSUR),* vol. 34, no. 1, pp. 1-47, 2002.

[19] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, pp. 888-905, 2000.

[20] G.W. Stewart and J.G. Sun, *Matrix Perturbation Theory.* Academic Press, 1990.

[21] V.N. Vapnik, *The Nature of Statistical Learning Theory.* Springer Verlag, 1995.

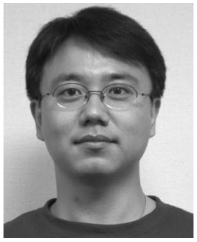
[22] V. Vural and J.G Dy, "A Hierarchical Method for Multi-Class Support Vector Machines," *Proc. Int'l Conf. Machine Learning,* 2004.

[23] J. Wang, H. Zeng, Z. Chen, H. Lu, L. Tao, and W. Ma, "ReCoM: Reinforcement Clustering of Multi-Type Interrelated Data Objects," *Proc. SIGIR '03,* 2003.

[24] Y. Yang, "A Scalability Analysis of Classifiers in Text Classification," *Proc. SIGIR '03,* 2003.

[25] H. Zha, C. Ding, and M. Gu, "Bipartite Graph Partitioning and Data Clustering," *Proc. Conf. Information and Knowledge Management,* 2001.

Bin Gao received the BS degree in mathematics from Shandong University, Jinan, China, in 2001. He is currently a PhD candidate at Peking University, Beijing, China. His research interests are in the areas of pattern recognition, machine learning, data mining, graph theory, and corresponding applications on text categorization and image processing.



Tie-Yan Liu received the BSc, MSc, and PhD degrees all from Tsinghua University in 1998, 2000, and 2003, respectively. After that, he joined Microsoft Research Asia as an associate researcher. His research interests include media content analysis, information retrieval, machine learning, and graph theory. He has had nearly 30 papers published in referred conferences and journals and has served on the technical program committees, workshop committees, or as a reviewer in a dozen of international conferences. He is a member of the IEEE Computer Society.

Guang Feng received the BS degree from Tsinghua University in 2003 and he is a PhD candidate at Tsinghua University. His research interests are in the areas of machine learning, information retrieval, graph representation, and complex network theory.

Tao Qin received the BS degree from Tsinghua University, Beijing, China, in 2003. He is currently a PhD candidate at Tsinghua University. From September 2003 to June 2004, he was with Microsoft Research Asia, Beijing, as a visiting student with the Media Computing Group. Since July 2004, he has been a visiting student with Web Search and Mining Group. His current interests include machine learning, information retrieval, graph representation and learning, and complex network theory.

Qian-Sheng Cheng received the BS degree in mathematics from Peking University, Beijing, China, in 1963. He is now a professor in the Department of Information Science, School of Mathematical Sciences, Peking University, China, and he was the vice director of the Institute of Mathematics, Peking University, from 1988 to 1996. His current research interests include signal processing, time series analysis, system identification, and pattern recognition. He is the vice chairman of the Chinese Signal Processing Society and has won the Third Chinese National Natural Science Award.

Wei-Ying Ma received the BS degree in electrical engineering from the National Tsing Hua University in Taiwan in 1990 and the MS and PhD degrees in electrical and computer engineering from the University of California at Santa Barbara (UCSB) in 1994 and 1997, respectively. From 1994 to 1997, he was engaged in the Alexandria Digital Library (ADL) project in UCSB while completing a PhD. From 1997 to 2001, he was with HP Labs, where he worked in the field of multimedia adaptation and distributed media services infrastructure for mobile Internet. He joined Microsoft Research Asia in April 2001 as the research manager of the Web Search and Mining Group, leading research in the areas of information retrieval, text mining, search, multimedia management, and mobile browsing. In 2003 and 2004, his research group published nine full papers in SIGIR, five full papers in WWW, and eight full papers in ACM Multimedia conference, which accounted for 5-10 percent of the total number of accepted papers in these conferences. He currently serves as an editor for the *ACM/Springer Multimedia Systems Journal* and associate editor for the *Journal of Multimedia Tools and Applications* published by Kluwer Academic Publishers. He has served on the organizing and program committees of many international conferences including ACM Multimedia, ACM SIGIR, ACM CIKM, WWW, ICME, CVPR, SPIE Multimedia Storage and Archiving Systems, SPIE Multimedia Communication and Networking, etc. He is also the general cochair of the International Multimedia Modeling (MMM) Conference 2005 and the International Conference on Image and Video Retrieval (CIVR) 2005. He is a member of the IEEE Computer Society.