Fast Rank-2 Nonnegative Matrix Factorization for Hierarchical Document Clustering

Da Kuang, Haesun Park
School of Computational Science and Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0765, USA
{da.kuang,hpark}@cc.gatech.edu

ABSTRACT

Nonnegative matrix factorization (NMF) has been successfully used as a clustering method especially for flat partitioning of documents. In this paper, we propose an efficient hierarchical document clustering method based on a new algorithm for rank-2 NMF. When the two block coordinate descent framework is applied to rank-2 NMF, each subproblem requires a solution for nonnegative least squares (NNLS) with only two columns. We design the algorithm for rank-2 NMF by exploiting the fact that an exhaustive search for the optimal active set can be performed extremely fast when solving these NNLS problems. In addition, we design a measure on the results of rank-2 NMF for determining which leaf node should be further split. On a number of text data sets, our proposed method produces high-quality tree structures in significantly less time compared to other methods such as hierarchical K-means, standard NMF, and latent Dirichlet allocation.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering*

General Terms

Algorithms, Experimentation, Performance

Keywords

Active-set algorithm, hierarchical document clustering, nonnegative matrix factorization, rank-2 NMF

1. INTRODUCTION

Nonnegative matrix factorization (NMF) has received wide recognition in many data mining areas such as text analysis [24]. In NMF, given a nonnegative matrix $X \in \mathbb{R}_+^{m\times n}$ and $k \leq \min(m, n)$, X is approximated by a product of two nonnegative matrices $W \in \mathbb{R}_+^{m\times k}$ and $H \in \mathbb{R}_+^{k\times n}$. A common way to define NMF is to use the Frobenius norm to measure the difference between X and WH [10]:

$$\min_{W,H\geq 0} \|X - WH\|_F^2 \tag{1}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. The columns of $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n]$ represent n nonnegative data points in the m-dimensional space. Typically $k << \min(m, n)$, i.e. the original data points in the m-dimensional space are approximated in a much lower-dimensional space of dimension k. The k columns of W are nonnegative basis vectors that span the lower-dimensional space, and the i-th column of H contains k nonnegative linear coefficients that represent $\mathbf{x}_i$ in the space spanned by the columns of W. Nonnegative data frequently occur in modern data analysis, such as text corpus when represented as a term-document matrix [18]. Because the lower rank factors W and H contain only nonnegative elements which stay in the original domain of the data points, NMF often produces basis vectors that facilitate better interpretation [14].

NMF has shown excellent performances as a clustering method in numerous applications [24, 5, 9]. When NMF is used as a clustering method, the columns of W are interpreted as k cluster representatives, and the i-th column of H contains fractional assignment values of the i-th data point for the k clusters, which can be interpreted as soft clustering. To obtain a hard clustering result for the i-th data point, we may choose the index that corresponds to the largest element in the i-th column of H. This clustering scheme has been shown to achieve superior clustering quality, and many variations such as constrained clustering and graph clustering [12, 9, 5, 16] have been proposed. The standard NMF (1) has been shown to perform especially well as a document clustering method, where the columns of W can be interpreted as k topics extracted from a corpus [24, 12]. When the j-th column of W is constrained to have unit L_1 norm, it can be interpreted as a probability distribution of terms for the j-th topic.

Most of the previous work on clustering with NMF has been focused on flat partitioning of a data set. However, hierarchical clustering often reveals additional structures in the data. For example, a tree structure often provides a more detailed taxonomy or a better description of natural phenomena than a flat partitioning. In the widely-used text corpus RCV1 [15], a hierarchy of topics was defined, with 103 leaf nodes under four super categories (Corporate/Industrial, Economics, Government/Social, Markets). Online retailers such as Amazon and eBay also maintain their product catalogues as a hierarchical tree structure. In this paper, we will explore hierarchical clustering with NMF and show its competitive clustering quality.

The lower rank k in standard NMF, which represents the number of clusters in a clustering setting, is often assumed to be given or predetermined according to prior knowledge about the data set or the embedding of the data points. Selecting the number of clusters k is an important and difficult issue in practice. Though model selection methods for selecting k have been proposed in the context of NMF [4], it is expensive to compute solutions of NMF for each k in general. In the NMF-based hierarchical approach we propose in this paper, a data set is recursively divided into small subsets and the number of clusters does not need to be predetermined by a user.

Hierarchical probabilistic topic modeling of document data sets, such as hierarchical latent Dirichlet allocation (hLDA) [2], is very popular in the community. We will show the conceptual difference between a clustering approach and probabilisitic modeling in a later section.

We will design a hierarchical clustering method based on rank-2 NMF, i.e. NMF with $k = 2$. The hierarchical structure we will generate is a binary tree, and our method does not require any input on the level of the tree or the total number of clusters. Our motivation for hierarchical clustering with binary tree structure is based on our fast algorithm for rank-2 NMF proposed in this paper. We will exploit the special properties of NMF with $k = 2$, and propose a very fast algorithm. We will study a particular type of existing algorithms for standard NMF, namely *active-set-type algorithms* [10, 11], and show that when $k = 2$, active-set-type algorithms can be reduced to a simple and efficient algorithm for rank-2 NMF, which has additional benefits when implemented on parallel platforms due to "non-random" memory access.

When applying rank-2 NMF to the recursive splitting of a text corpus, we need to automatically determine the next node to split. Our strategy is to collect rank-2 NMF results on all the current leaf nodes before deciding which one to split, and compute a score for each leaf node to evaluate whether it is composed of two well-separated clusters based on the two basis vectors generated by rank-2 NMF. Compared to existing strategies that rely on an $n \times n$ document-document similarity matrix [6], our methodology never generates a large dense matrix thus is more time/space efficient. Although the rank-2 NMF computation on any leaf node in the final tree is wasted, our methodology is still very efficient overall due to the high efficiency of our rank-2 NMF algorithm.

Our contributions in this paper include:

- We propose an active-set-type algorithm for rank-2 NMF, which is fast, guaranteed to converge, and easy to parallelize.
- By combining rank-2 NMF with a designed scoring function for every leaf node, we develop an efficient workflow for hierarchical document clustering with outlier detection. Our methodology is able to determine both the tree structure and the depth of the tree on-the-fly, in contrast to hierarchical probabilistic modeling methods that require the depth of the tree be specified by the user.
- We present promising empirical results of our methodology in terms of efficiency, clustering quality, as well as semantic quality in the topic modeling context. To the best of our knowledge, our work is the first attempt to cluster the full RCV1 data set [15] which contains approximately 800,000 documents. Our method finished in about 7 minutes on a shared-memory machine with two quad core CPUs and achieved better quality than standard NMF which costs 6.5 hours.

The rest of the paper is organized as follows. We conduct detailed analysis of existing active-set-type algorithms for NMF in the special case of $k = 2$ in Section 2, and present our new algorithm for rank-2 NMF in Section 3. In Section 4, we describe our measure for scoring tree nodes and the hierarchical document clustering workflow. In Section 5, we show the difference between clustering approaches and topic modeling approaches when applied to flat and hierarchical document clustering. In Section 6, we demonstrate the promising efficiency, clustering quality, and semantic quality of our methodology empirically on large-scale data sets. In Section 7, we summarize the advantages and shortcomings of this work. Although we focus on document clustering, the proposed hierarchical clustering method is not limited to documents.

Throughout this paper, $\|\cdot\|$ denotes the Euclidean norm, and $\|\cdot\|_F$ denotes the Frobenius norm.

2. ALTERNATING NONNEGATIVE LEAST SQUARES FOR NMF

In this paper, we consider the algorithms for NMF that fit into the two-block coordinate descent framework [17, 10, 11] due to better theoretical guarantee in convergence properties. In this framework, starting from some initialization, the matrices W and H are updated in an iterative manner, until some stopping criterion is satisfied. The overall nonconvex problem (1) is thus reduced to two convex subproblems:

$$\min_{W \geq 0} \quad \|H^T W^T - X^T\|_F^2 \tag{2}$$

$$\min_{H \geq 0} \quad \|WH - X\|_F^2 \tag{3}$$

When an optimal solution is obtained for each subproblem in each iteration, this iterative procedure is guaranteed to converge to a stationary point [7], which is an excellent convergence property for nonconvex problems such as (1). Each subproblem is a nonnegative least squares problem (NNLS) with multiple right-hand sides. Consider the following generic form for simplicity:

$$\min_{G \geq 0} \|BG - Y\|_F^2 \tag{4}$$

where $B \in \mathbb{R}_+^{m \times k}, Y \in \mathbb{R}_+^{m \times n}$ are given, and $G \in \mathbb{R}_+^{k \times n}$ is to be solved.

Various types of algorithms can be used to solve the NNLS problem and can be categorized into standard optimization algorithms and active-set-type algorithms. A classical active-set algorithm for NNLS with a single right-hand side was introduced in [13]. In the context of NMF, Lin [17] claimed that it would be expensive to solve NNLS with multiple right-hand sides using the active-set algorithm repeatedly, and proposed a projected gradient descent (PGD) algorithm. However, Kim & Park [10] proposed several improvements for the original active-set algorithm, and achieved an NMF algorithm with overall efficiency comparable to PGD.

Later, a block-pivoting algorithm for NNLS [11] was proposed, which proved to be more efficient than the active-set algorithm when k is large. We call both active-set based and block-pivoting based algorithms for NMF as *active-set-type algorithms*.

In active-set-type algorithms for NNLS, we need to identify a partitioning of variables in G into an *active set* $\mathcal{A}$ and a *passive set* $\mathcal{P}$. At each step of searching for the optimal active set, we need to solve an unconstrained least squares problem. Because the number of possible active sets is exponential, a well-guided search of the optimal active sets is important, such as presented by the active-set and block-pivoting methods. To improve the efficiency of solving NNLS with multiple right-hand sides (4), the columns of G with the same active set pattern are grouped together for lower computational complexity and more cache-efficient computation [10, 22], and the grouping of columns changes when the active set is re-identified in each iteration of NNLS. Practically, the grouping step is implemented as a sorting of the columns of G, with complexity $O(n \log n)$ which is expensive when n is large. Other steps, such as checking the optimality of the active sets, also introduces additional overheads.

When the underlying application restricts the value of k to be 2, such as hierarchical clustering that generates a binary tree, the number of possible active sets is reduced to $2^2 = 4$ for each column of G, and it is practical to enumerate all of them. Conceptually, active-set-type algorithms search for the optimal active set in a finite set of possible active sets, and the size of the finite search space is 4 in the special case of $k = 2$. On the contrary, standard optimization algorithms require an indefinite number of search steps before convergence, and the actual number of iterations depends on the required accuracy. Therefore, when $k = 2$, standard optimization algorithms such as PGD are not able to exploit the special property of NNLS, and active-set-type algorithms become the better choice. In the next section, we will propose a new algorithm and its efficient implementation based on active-set-type algorithms, which will avoid all the overheads of switching between active sets.

Both flat clustering based on standard NMF and hierarchical clustering based on rank-2 NMF can produce k non-overlapping groups of a data set. In the following, we argue that the hierarchical approach is the preferred choice in terms of efficiency by conducting an analysis of computational complexity under different k values, using the active-set based algorithm [10] as an exemplar algorithm. Given an $m \times n$ data matrix and the number of clusters k, the complexity of one NNLS run for standard NMF is:

$$4mnk + 2(m+n)k^2 + t[(1/3)k^3 + 2(m+n)k^2] \text{ flops} \quad (5)$$

where t is the number of search steps towards the optimal active set. In hierarchical clustering, we need to perform rank-2 NMF for $k-1$ times, and the complexity of one NNLS run summed over all the $k-1$ steps is at most:

$$(k-1) \cdot [8mn + 8(m+n) + t(8/3 + 8m + 8n)] \text{ flops} \quad (6)$$

The actual flops (floating point operations) in hierarchical clustering must be smaller than (6), because any splitting other than the first step is executed on a subset of the data set only. Thus, the expression (5) is superlinear with respect to k, while (6) is linear with respect to k. Assuming the number of search steps t is the same in both cases, the hierarchical approach is expected to be much less expensive. With our new algorithm specifically for rank-2 NMF in this paper, the efficiency of NMF-based hierarchical clustering will be boosted even more.

3. A FAST ALGORITHM FOR $\min_{G \geq 0} \|BG - Y\|_F^2$ WITH $B \in \mathbb{R}_+^{M \times 2}, Y \geq 0$

We have reduced the problem of solving NMF to the problem of solving NNLS with multiple right-hand sides: $\min_{G \geq 0} \|BG - Y\|_F^2$. In the context of NMF, Y is set to either the data matrix X, or X^T. Let $Y = [\mathbf{y}_1, \cdots, \mathbf{y}_n], G = [\mathbf{g}_1, \cdots, \mathbf{g}_n]$. We emphasize that $\mathbf{y}_i \geq 0$ $(1 \leq i \leq n)$ in the NNLS problem we are solving, since the data is nonnegative.

Since the formulation (4) for NNLS with multiple right-hand sides can be rewritten as

$$\min_{\mathbf{g}_1, \cdots, \mathbf{g}_n \geq 0} \|B\mathbf{g}_1 - \mathbf{y}_1\|^2 + \cdots + \|B\mathbf{g}_n - \mathbf{y}_n\|^2 \quad (7)$$

the solution of each column of G is independent of each other, and we obtain n NNLS problems each with a single right-hand side. We first study the solution of NNLS with a single right-hand side, and then consider the issues when combining multiple right-hand sides.

3.1 Solution of $\min_{\mathbf{g} \geq 0} \|B\mathbf{g} - \mathbf{y}\|$

In general, when B has more than two columns, an active-set-type algorithm has to search for an optimal active set as discussed in Section 2. We denote the two parts of $\mathbf{g}$ that correspond to the active set and the passive set as $\mathbf{g}_{\mathcal{A}}$ and $\mathbf{g}_{\mathcal{P}}$, respectively. At each iteration of the algorithm, $\mathbf{g}_{\mathcal{A}}$ is set to zero, and $\mathbf{g}_{\mathcal{P}}$ is solved by unconstrained least squares using a subset of columns of B corresponding to $\mathcal{P}$. The optimal active set is the one where the solution of unconstrained least squares is feasible, i.e. $\mathbf{g}_{\mathcal{P}} \geq 0$, and meanwhile $\|B\mathbf{g} - \mathbf{y}\|^2$ is minimized.

When $k = 2$, we have

$$J(\mathbf{g}) \equiv \|B\mathbf{g} - \mathbf{y}\|^2 = \|\mathbf{b_1} g_1 + \mathbf{b_2} g_2 - \mathbf{y}\|^2 \quad (8)$$

where $B = [\mathbf{b}_1, \mathbf{b}_2] \in \mathbb{R}_+^{m \times 2}$, $\mathbf{y} \in \mathbb{R}_+^{m \times 1}$, and $\mathbf{g} = [g_1, g_2]^T \in \mathbb{R}^{2 \times 1}$.

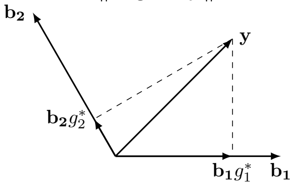
Considering the limited number of possible active sets, our idea is to avoid the search of the optimal active set at the cost of some redundant computation. The four possibilities of the active set $\mathcal{A}$ is shown in Table 1. We simply enumerate all the possibilities of $(\mathcal{A}, \mathcal{P})$, and for each $\mathcal{P}$, minimize the corresponding objective function $J(\mathbf{g})$ in Table 1 by solving unconstrained least squares. Then, of all the feasible solutions of $\mathbf{g}$ (i.e. $\mathbf{g} \geq 0$), we pick the one with the smallest $J(\mathbf{g})$. Now we study the properties of the solutions of these unconstrained least squares problems, which will lead to an efficient algorithm to find the optimal active set.

Table 1: Four possible active sets when $B \in \mathbb{R}_+^{m \times 2}$.

$\mathcal{A}$	$\mathcal{P}$	$J(\mathbf{g})$
$\{1,2\}$	$\emptyset$	$\|\mathbf{y}\|^2$
$\{1\}$	$\{2\}$	$\|\mathbf{b_2} g_2 - \mathbf{y}\|^2$
$\{2\}$	$\{1\}$	$\|\mathbf{b_1} g_1 - \mathbf{y}\|^2$
$\emptyset$	$\{1,2\}$	$\|\mathbf{b_1} g_1 + \mathbf{b_2} g_2 - \mathbf{y}\|^2$

First, we claim that the two unconstrained problems min $\|\mathbf{b_1} g_1 - \mathbf{y}\|^2$, min $\|\mathbf{b_2} g_2 - \mathbf{y}\|^2$ always yield feasible solutions.

Figure 1: An illustration of one-dimensional least squares problems $\min \|\mathbf{b}_1 g_1^* - \mathbf{y}\|^2$ **and** $\min \|\mathbf{b}_2 g_2^* - \mathbf{y}\|^2$**.**



Algorithm 1 Algorithm for solving $\min_{\mathbf{g}\geq 0} \|B\mathbf{g} - \mathbf{y}\|^2$, where $B = [\mathbf{b}_1, \mathbf{b}_2] \in \mathbb{R}_+^{m\times 2}, \mathbf{y} \in \mathbb{R}_+^{m\times 1}$

```
 1: Solve unconstrained least squares $\mathbf{g}^\emptyset \leftarrow \min \|B\mathbf{g} - \mathbf{y}\|^2$
 2: if $\mathbf{g}^\emptyset \geq 0$ then
 3:   return $\mathbf{g}^\emptyset$
 4: else
 5:   $g_1^* \leftarrow (\mathbf{y}^T\mathbf{b}_1)/(\mathbf{b}_1^T\mathbf{b}_1)$
 6:   $g_2^* \leftarrow (\mathbf{y}^T\mathbf{b}_2)/(\mathbf{b}_2^T\mathbf{b}_2)$
 7:   if $g_1^*\|\mathbf{b}_1\| \geq g_2^*\|\mathbf{b}_2\|$ then
 8:     return $[g_1^*, 0]^T$
 9:   else
10:     return $[0, g_2^*]^T$
11:   end if
12: end if
```

Take $\min \|\mathbf{b}_1 g_1 - \mathbf{y}\|^2$ as an example. Its solution is:

$$g_1^* = \frac{\mathbf{y}^T\mathbf{b}_1}{\mathbf{b}_1^T\mathbf{b}_1} \tag{9}$$

If $\mathbf{b}_1 \neq 0$, we always have $g_1^* \geq 0$ since $\mathbf{y} \geq 0, \mathbf{b}_1 \geq 0$. In the context of rank-2 NMF, the columns of W and the rows of H are usually linearly independent when nonnegative-rank(X) ≥ 2, thus $\mathbf{b}_1 \neq 0$ holds in most cases. Geometrically (see Fig. 1), the best approximation of vector $\mathbf{y}$ in the one-dimensional space spanned by $\mathbf{b}_1$ is the orthogonal projection of $\mathbf{y}$ onto $\mathbf{b}_1$.

If $\mathbf{g}^\emptyset \equiv \arg\min \|\mathbf{b}_1 g_1 + \mathbf{b}_2 g_2 - \mathbf{y}\|^2$ is nonnegative, then $\mathcal{A} = \emptyset$ is the optimal active set because the unconstrained solution $\mathbf{g}^\emptyset$ is feasible and neither $\min \|\mathbf{b}_1 g_1 - \mathbf{y}_1\|^2$ nor $\min \|\mathbf{b}_2 g_2 - \mathbf{y}_2\|^2$ can be smaller than $J(\mathbf{g}^\emptyset)$. Otherwise, we only need to find the smallest objective $J(\mathbf{g})$ among the other three cases since they all yield feasible solutions. We claim that $\mathcal{A} = \{1, 2\}$, i.e. $\mathcal{P} = \emptyset$, can be excluded. Using g_1^*, the solution of $\min \|\mathbf{b}_1 g_1 - \mathbf{y}\|^2$, we have

$$\|\mathbf{b}_1 g_1^* - \mathbf{y}\|^2 = \|\mathbf{y}\|^2 - (\mathbf{y}^T\mathbf{b}_1)^2/(\mathbf{b}_1^T\mathbf{b}_1) \leq \|\mathbf{y}\|^2 \tag{10}$$

In fact, $\mathcal{P} = \{1\}$ includes $\mathcal{P} = \emptyset$ as a special case when $\mathbf{b}_1 \perp \mathbf{y}$.

To compare $\|\mathbf{b}_1 g_1^* - \mathbf{y}\|^2$ and $\|\mathbf{b}_2 g_2^* - \mathbf{y}\|^2$, we note that in the illustration in Fig. 1, $\mathbf{b}_1 g_1^* - \mathbf{y} \perp \mathbf{b}_1 g_1^*$ and $\mathbf{b}_2 g_2^* - \mathbf{y} \perp \mathbf{b}_2 g_2^*$, therefore we have:

$$\|\mathbf{b}_j g_j^*\|^2 + \|\mathbf{b}_j g_j^* - \mathbf{y}\|^2 = \|\mathbf{y}\|^2 \tag{11}$$

for $j = 1, 2$. Thus choosing the smaller objective amounts to choosing the larger value from $g_1^*\|\mathbf{b}_1\|$ and $g_2^*\|\mathbf{b}_2\|$.

Our algorithm for NNLS with a single right-hand side is summarized in Algorithm 1.

3.2 Solution of $\min_{G\geq 0} \|BG - Y\|_F$

When Algorithm 1 is applied to NNLS with multiple right-hand sides, computing $\mathbf{g}^\emptyset, g_1^*, g_2^*$ for different vectors $\mathbf{y}$ separately is not cache-efficient. In Algorithm 2, we solve NNLS with n different vectors $\mathbf{y}$ simultaneously, and the analysis in Section 3.1 becomes important. Note that the entire for-loop (line 5-15, Algorithm 2) is embarrassingly parallel and can be vectorized. To achieve this, unconstrained solutions for all the three possible active sets are computed before entering the for-loop. Some computation is redundant, for example, the cost of solving u_i and v_i is wasted when $\mathbf{g}_i^\emptyset \geq 0$ (c.f. line 5-6, Algorithm 1). However, Algorithm 2 represents a non-random pattern of memory access, and we expect that it is much faster for rank-2 NMF than applying existing active-set-type algorithms directly.

Algorithm 2 Algorithm for solving $\min_{G\geq 0} \|BG - Y\|_F^2$, where $B = [\mathbf{b}_1, \mathbf{b}_2] \in \mathbb{R}_+^{m\times 2}, Y \in \mathbb{R}_+^{m\times n}$

```
 1: Solve unconstrained least squares $G^\emptyset = [\mathbf{g}_1^\emptyset, \cdots, \mathbf{g}_n^\emptyset] \leftarrow$
    $\min \|BG - Y\|^2$
 2: $\beta_1 \leftarrow \|\mathbf{b}_1\|$, $\beta_2 \leftarrow \|\mathbf{b}_2\|$
 3: $\mathbf{u} \leftarrow (Y^T\mathbf{b}_1)/\beta_1^2$
 4: $\mathbf{v} \leftarrow (Y^T\mathbf{b}_2)/\beta_2^2$
 5: for $i = 1$ to $n$ do
 6:   if $\mathbf{g}_i^\emptyset \geq 0$ then
 7:     return $\mathbf{g}_i^\emptyset$
 8:   else
 9:     if $u_i\beta_1 \geq v_i\beta_2$ then
10:       return $[u_i, 0]^T$
11:     else
12:       return $[0, v_i]^T$
13:     end if
14:   end if
15: end for
```

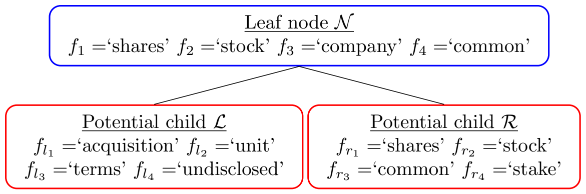
Note that a naïve implementation of comparing $\|\mathbf{b}_1 g_1^* - \mathbf{y}\|^2$ and $\|\mathbf{b}_2 g_2^* - \mathbf{y}\|^2$ for n different vectors $\mathbf{y}$ requires $O(mn)$ complexity due to the creation of the $m \times n$ dense matrix $BG - Y$. In contrast, our algorithm only requires $O(m + n)$ complexity at this step (line 9, Algorithm 2), because $\mathbf{b}_1, \mathbf{b}_2$ are the same across all the n right-hand sides. The overall complexity of Algorithm 2 is still $O(mn)$, though, which is the same as the complexity of existing active-set-type algorithms when $k = 2$ (see Eq. 5). The dominant part comes from computing the matrix product Y^TB in unconstrained least squares.

4. HIERARCHICAL DOCUMENT CLUSTERING BASED ON RANK-2 NMF

Rank-2 NMF can be recursively applied to a data set, generating a hierarchical tree structure of items. In this section, we focus on text corpus and develop an overall efficient approach to hierarchical document clustering with outlier detection. In particular, we need to have a strategy of choosing an existing leaf node at each splitting step. Extensive criteria for selecting the next leaf node to split were discussed in previous literature for general clustering methods [6], mainly relying on cluster labels induced by the current tree structure. In the context of NMF, however, we have additional information about the clusters: Each column of W is a cluster representative. In text data, a column of W is the term distribution for a topic [24], and the largest elements in the column correspond to the *top words* for this topic. We will exploit this information to determine the next node to split.

In summary, our strategy is to compute a score for each

Figure 2: An illustration of a leaf node $\mathcal{N}$ and its two potential children $\mathcal{L}$ and $\mathcal{R}$.



Leaf node $\mathcal{N}$
f_1 ='shares' f_2 ='stock' f_3 ='company' f_4 ='common'
Potential child $\mathcal{L}$
f_{l_1} ='acquisition' f_{l_2} ='unit'
f_{l_3} ='terms' f_{l_4} ='undisclosed'
Potential child $\mathcal{R}$
f_{r_1} ='shares' f_{r_2} ='stock'
f_{r_3} ='common' f_{r_4} ='stake'

leaf node, by running rank-2 NMF on this node and evaluating the two columns of W. Then we select the current leaf node with the highest score as the next node to split. The score for each node needs only to be computed once when the node first appears in the tree. For an illustration of a leaf node and its two potential children, see Fig. 2. A leaf node $\mathcal{N}$ should be split if at least two well-separated topics can be discovered within the node. Thus we expect that $\mathcal{N}$ receives a high score if the top words for $\mathcal{N}$ is a good combination of the top words for its two potential children, $\mathcal{L}$ and $\mathcal{R}$. We also expect that $\mathcal{N}$ receives a low score if the top words for $\mathcal{L}$ and $\mathcal{R}$ are almost the same.

To be precise, we borrow the concept of normalized discounted cumulative gain (NDCG) [8] from the information retrieval community. Given a perfect ranked list, NDCG measures the quality of an actual ranked list which always has value between 0 and 1. A leaf node $\mathcal{N}$ in our tree is associated with a term distribution $w_\mathcal{N}$, given by a column of W from the rank-2 NMF run that generates the node $\mathcal{N}$. We can obtain a ranked list of terms for $\mathcal{N}$, by sorting the elements in $w_\mathcal{N}$ in descending order, denoted by $f_\mathcal{N}$. Similarly, we can obtain ranked lists of terms for its two potential children, $\mathcal{L}$ and $\mathcal{R}$, denoted by $f_\mathcal{L}$ and $f_\mathcal{R}$. Assuming $f_\mathcal{N}$ is a perfect ranked list, we compute a modified NDCG (mNDCG) score for each of $f_\mathcal{L}$ and $f_\mathcal{R}$. We describe our way to compute mNDCG in the following. Recall that m is the total number of terms in the vocabulary. Suppose the perfectly ordered terms corresponding to $f_\mathcal{N}$ is

$$f_1, f_2, \cdots, f_m$$

and the shuffled orderings in $f_\mathcal{L}$ and $f_\mathcal{R}$ are respectively:

$$\begin{array}{c} f_{l_1}, f_{l_2}, \cdots, f_{l_m} \\ f_{r_1}, f_{r_2}, \cdots, f_{r_m} \end{array}$$

We first define a *position discount factor* $p(f_i)$ and a *gain* $g(f_i)$ for each term f_i:

$$\begin{aligned} p(f_i) &= \log\left(m - \max\{i_1, i_2\} + 1\right) && (12) \\ g(f_i) &= \frac{\log(m - i + 1)}{p(f_i)} && (13) \end{aligned}$$

where $l_{i_1} = r_{i_2} = i$. In other words, for each term f_i, we find its positions i_1, i_2 in the two shuffled orderings, and place a large discount in the gain of term f_i if this term is high-ranked in both shuffled orderings. The sequence of gain $\{g(f_i)\}_{i=1}^m$ is sorted in descending order, resulting in another sequence $\{\hat{g}_i\}_{i=1}^m$. Then, for a shuffled ordering $f_\mathcal{S}$ ($f_\mathcal{S} = f_\mathcal{L}$ or $f_\mathcal{R}$), mNDCG is defined as:

$$\begin{aligned} \text{mDCG}(f_\mathcal{S}) &= g(f_{s_1}) + \sum_{i=2}^{m} \frac{g(f_{s_i})}{\log_2(i)} && (14) \\ \text{mIDCG} &= \hat{g}_1 + \sum_{i=2}^{m} \frac{\hat{g}_i}{\log_2(i)} && (15) \\ \text{mNDCG}(f_\mathcal{S}) &= \frac{\text{mDCG}(f_\mathcal{S})}{\text{mIDCG}} && (16) \end{aligned}$$

As we can see, mNDCG is basically computed in the same way as the standard NDCG measure, but with a modified gain function. Also note that $\hat{g}_i$ instead of $g(f_i)$ is used in computing the ideal mDCG (mIDCG) so that mNDCG always has a value in the $[0, 1]$ interval.

Finally, the score of the leaf node $\mathcal{N}$ is computed as:

$$\text{score}(\mathcal{N}) = \text{mNDCG}(f_\mathcal{L}) \times \text{mNDCG}(f_\mathcal{R}) \quad (17)$$

To illustrate the effectiveness of this scoring function, let us consider some special cases.

1. When the two potential children $\mathcal{L}, \mathcal{R}$ describe well-separated topics, a top word for $\mathcal{N}$ is high-ranked in one of the two shuffled orderings $f_\mathcal{L}, f_\mathcal{R}$, and low-ranked in the other. Thus the top words will not suffer from a large discount, and both mNDCG$(f_\mathcal{L})$ and mNDCG$(f_\mathcal{R})$ will be large.

2. When both $\mathcal{L}$ and $\mathcal{R}$ describe the same topic as that of $\mathcal{N}$, a top word for $\mathcal{N}$ is high-ranked in both the shuffled orderings. Thus the top words will get a large discount, and both mNDCG$(f_\mathcal{L})$ and mNDCG$(f_\mathcal{R})$ will be small.

3. When $\mathcal{L}$ describes the same topic as that of $\mathcal{N}$, and $\mathcal{R}$ describes a totally unrelated topic (e.g. outliers in $\mathcal{N}$), then mNDCG$(f_\mathcal{L})$ is large and mNDCG$(f_\mathcal{R})$ is small, and score$(\mathcal{N})$ is small.

The overall hierarchical document clustering workflow is summarized in Algorithm 3, where we refer to a node and the documents associated with the node exchangably. The while-loop in this workflow (line 8-15) defines an outlier detection procedure, where T trials of rank-2 NMF are allowed in order to split a leaf node $\mathcal{M}$ into two well-separated clusters. At each trial, two potential children nodes $\mathcal{N}_1, \mathcal{N}_2$ are created, and if we believe that one of them (say, $\mathcal{N}_2$) is composed of outliers, we discard $\mathcal{N}_2$ from $\mathcal{M}$ at the next trial. If we still cannot split $\mathcal{M}$ into two well-separated clusters after T trials, $\mathcal{M}$ is marked as a permanent leaf node. Thus Algorithm 3 has also described when to stop splitting a certain node. We have not specified the best moment to exit and stop the recursive splitting process, but simply set an upper limit of leaf nodes k. Other strategies can be used to determine when to exit, such as specifying a score threshold σ and exiting the program when none of the leaf nodes have scores above σ; $\sigma = 0$ means that the recursive splitting process is not finished until all the leaf nodes become permanent leaf nodes.

Compared to other criteria for choosing the next node to split, such as those relying on the self-similarity of each cluster and incurring $O(n^2)$ overhead [6], our method is more efficient. In practice, the binary tree structure that results from Algorithm 3 often has meaningful hierarchies and leaf clusters. We will evaluate its performance by standard measures in the experiment section.

Algorithm 3 Hierarchical document clustering based on rank-2 NMF

```
1: Input: A term-document matrix $X \in \mathbb{R}_+^{m\times n}$ (often
   sparse), maximum number of leaf nodes $k$, parameter
   $\beta > 1$ and $T \in \mathbb{N}$ for outlier detection
2: Create a root node $\mathcal{R}$, containing all the $n$ documents
3: score($\mathcal{R}$) $\leftarrow \infty$
4: repeat
5:    $\mathcal{M} \leftarrow$ a current leaf node with the highest score
6:    Trial index $i \leftarrow 0$
7:    Outlier set $Z \leftarrow \emptyset$
8:    while $i < T$ do
9:       Run rank-2 NMF on $\mathcal{M}$ and create two potential
         children $\mathcal{N}_1, \mathcal{N}_2$, where $|\mathcal{N}_1| \geq |\mathcal{N}_2|$
10:      if $|\mathcal{N}_1| \geq \beta|\mathcal{N}_2|$ and score($\mathcal{N}_2$) is smaller than every
         positive score of current leaf nodes then
11:         $Z \leftarrow Z \cup \mathcal{N}_2$, $\mathcal{M} \leftarrow \mathcal{M} - Z$, $i \leftarrow i + 1$
12:      else
13:         break
14:      end if
15:   end while
16:   if $i < T$ then
17:      Split $\mathcal{M}$ into $\mathcal{N}_1$ and $\mathcal{N}_2$
18:      Compute score($\mathcal{N}_1$) and score($\mathcal{N}_2$)
19:   else
20:      $\mathcal{M} \leftarrow \mathcal{M} \cup Z$ (recycle the outliers and do not split
         $\mathcal{M}$)
21:      score($M$) $\leftarrow -1$ (set $\mathcal{M}$ as a permanent leaf node)
22:   end if
23: until # leaf nodes = $k$
24: Output: A binary tree structure of documents, where
    each node has a ranked list of terms
```

5. RELATED WORK

NMF can be regarded as both a clustering method and, under certain constraints, a probabilistic topic modeling method. Both document clustering and topic modeling can be thought of as some sort of dimension reduction, because they find a set of latent topics as term distributions (columns of W) as well as topic proportions for each document (columns of H). NMF has a long history in document clustering [24], and its formulation has been used to compute topic models recently [1]. However, these two tasks have fundamental differences. Our paper is focused on the clustering aspect, and now we compare NMF-based clustering to a popular topic modeling method, latent Dirichlet allocation (LDA) [3].

We start with comparing flat clustering and flat topic modeling. First, LDA builds a probabilistic model that generates the text corpus, and intends to predict the probability of new documents. The model should not overfit the text corpus currently available, and model selection is usually done via cross-validation. On the contrary, NMF-based clustering is devoted to the current text corpus only, and its goal is to derive a partitioning that well-organizes the corpus. Second, LDA models each word as a discrete random variable, thus is not compatible with tf-idf weighting when forming the term-document matrix. However, NMF-based clustering finds an algebraic latent subspace and is able to leverage the benefit of tf-idf weighting which has proved to be useful in a wide range of tasks such as information retrieval [18]. Finally, the number of topics specified, when applying LDA, is often set to $100 \sim 400$ [3]; whereas the number of clusters is often much smaller, and resulting clusters describe much higher-level topics. We generate at most 70 clusters for any corpus in our experiments.

Now we discuss the difference between hierarchical clustering based on rank-2 NMF and hierarchical LDA (hLDA) [2]. hLDA builds a hierarchy of topics and each document is generated by sampling from the topics along a path with length L from the root to a leaf node. On the contrary, hierarchical clustering builds a hierarchy of documents, and the documents associated with each node are a mixture of two topics extracted from this node. In practice, hLDA requires all the leaf nodes be on the same level of the tree, and the depth L of the tree is chosen beforehand, while hierarchical clustering adaptively chooses a node at each splitting step.

In general, both methods have pros and cons: Topic modeling has a probabilistic interpretation, and clustering approaches are more flexible. In the next section, we include LDA in our experiments and compare its performance with the performances of clustering based on NMF and rank-2 NMF. hLDA or other hierarchical probabilistic models are not considered in the experiments because they represent a quite different scenario of text modeling, for example, the number of children of each node cannot be given as input [2].

As a final note, compared to previous divisive hierarchical clustering algorithms [6, 25, 21], our methodology splits only the nodes that consist of well-separated clusters and does not require additional merging steps. In the experiments, we will demonstrate the efficiency of our method by applying it to the full RCV1 data set [15] rather than extracting a small subset as seen in previous works.

6. EXPERIMENTS

In this section, we describe our experimental settings and demonstrate both the efficiency and quality of our proposed algorithm. All the experiments except LDA are run in Matlab 7.9 (R2009b) with two Intel Xeon X5550 quad-core processors and 24GB memory.

6.1 Data Sets

Four text data sets with ground-truth classes are used in our experiments: 1. **Reuters-21578**[1] contains news articles from the Reuters newswire in 1987. We discarded documents with multiple class labels, and then selected the 20 largest classes. 2. **20 Newsgroups**[2] contains articles from Usenet newsgroups and has a defined hierarchy of 3 levels. Unlike previous indexing, we observed that many articles have duplicated paragraphs due to cross-referencing. We discarded cited paragraphs and signatures. 3. **Cora** [19] is a collection of research papers in computer science, from which we extracted the title, abstract, and reference-contexts. Although this data set defines a topic hierarchy of 3 levels, we observed that some topics, such as "AI – NLP" and "IR – Extraction", are very related but reside in different subtrees. Thus we ignored the hierarchy and obtained 70 ground-truth classes as a flat partitioning. 4. **RCV1** [15] is a much larger collection of news articles from Reuters. It contains over 800,000 articles in the time period of 1996-1997 and defines a sophisticated topic hierarchy with 103

[1]`http://www.daviddlewis.com/resources/testcollections/reuters21578/`
[2]`http://qwone.com/~jason/20Newsgroups/`

labels. We discarded documents with multiple class labels, and then selected the 40 largest classes, named as **RCV1-labeled**. The full data set, named as **RCV1-full**, is also included in our experiments with no ground-truth classes.

We summarize these data sets in Table 2. All the data sets except 20 Newsgroups have very unbalanced sizes of ground-truth classes.

6.2 Methods for Comparison

The clustering methods in our experiments are named as follows:

- `r2-nmf-hier`: Hierarchical clustering based on rank-2 NMF with the algorithm proposed in this paper.
- `nmf-hier`: Hierarchical clustering based on standard NMF with active-set based algorithm [10]. In our experiments, multiplicative update rule algorithms [14] for standard NMF are always slower and give similar quality compared to active-set-type algorithms, thus are not included in our results.
- `nmf-flat`: Flat clustering based on standard NMF with block-pivoting based algorithm [11].
- `kmeans-hier`: Hierarchical clustering based on standard K-means. We use the hierarchical clustering workflow described in Algorithm 3; however, the term distribution associated with each node is given by the centroid vector from the K-means run that generates this node.
- `kmeans-flat`: Flat clustering based on standard K-means.
- `lda`: Flat clustering using the Gibbs sampling algorithm for LDA. We use a highly-optimized implementation in the software MALLET[3] written in Java. LDA is not run for RCV1-labeled and RCV1-full due to efficiency reasons.

Hierarchical clustering and flat clustering cannot be compared against each other directly. We evaluate the hierarchy by taking snapshots of the tree as leaf nodes are generated, and because leaf nodes are non-overlapping, we treat all the leaf nodes in each snapshot as a flat partitioning. Thus, if the maximum number of leaf nodes is c, we produce $c-1$ flat partitionings forming a hierarchy.

For each method, we perform 20 runs on medium-scale data sets and 5 runs on large-scale data sets starting from random initializations. Average measurements are reported. Note that for flat clustering methods, each run consists of $c-1$ separate executions with the number of clusters set to $2, 3, \cdots, c$.

The maximum number of leaf nodes c is set to be the number of ground-truth classes at the deepest level (see Table 2). The hierarchical clustering workflow (Algorithm 3) runs with parameters $\beta = 9$, $T = 3$ (for `r2-nmf-hier` and `nmf-hier`) or 5 (for `kmeans-hier`). The K-means implementation is our optimized version of the Matlab `kmeans` function, which has a batch-update phase and a more time-consuming online-update phase. We use both phases for medium-scale data sets and only the batch-update phase for large-scale data sets. Every method is implemented with multi-threading.

[3] `http://mallet.cs.umass.edu/`

6.3 Evaluation Measures

Each of the six methods described above can be regarded as both a clustering method and a topic modeling method. We use the following two measures to evaluate their quality:

1. *Normalized mutual information (NMI)*: This is a measure of the similarity between two flat partitionings. It is used to evaluate clustering quality and is only applicable to data sets with ground-truth classes. It is particularly useful when the number of generated clusters is different from that of ground-truth classes and can be used to determine the optimal number of clusters. More details can be found in [18]. For data sets with defined hierarchy, we compute NMI between a generated partitioning and the ground-truth classes at each level of the tree; if the tree has depth L, then we compute L measures corresponding to each level.

2. *Coherence*: This is a measure of intra-topic similarity in topic models [20, 1]. Given the top words $f_1, \cdots, f_K$ for a topic, coherence is computed as

$$\text{coherence} = \sum_{i=1}^{K} \sum_{j=i}^{K} \left(\log \frac{D(f_i, f_j) + \epsilon}{D(f_i)} \right) \tag{18}$$

where $D(f_i)$ is the document frequency of f_i. $D(f_i, f_j)$ is the number of documents that contain both f_1 and f_2, and ϵ is a smoothing parameter. We use $\epsilon = 1$ and $K = 20$ [20]. The coherence averaged over all the topics is reported.

6.4 Timing Results

Timing results of the six methods are shown in Fig. 3. Hierarchical clustering based on rank-2 NMF is much faster than flat clustering using NMF or LDA. These results have verified our complexity analysis in Section 2, that flat clustering based on standard NMF exhibits a superlinear trend while hierarchical clustering based on rank-2 NMF exhibits a linear trend of running time as k increases. The first two plots correspond to medium-scale data sets, and `r2-nmf-hier` only requires about 1/3 the time needed by `nmf-hier`. The other three plots correspond to large-scale data sets, where we use logarithmic scale. K-means with only the batch-update phase also runs fast; however, their clustering quality is not as good, which will be shown later.

The difference between our proposed algorithm and the original active-set based algorithm for rank-2 NMF is less substantial as the data size increases. The performance is mainly bounded by the computation of $Y^T B$ in Algorithm 2. Because $B \in \mathbb{R}_+^{m \times 2}$ is a very long-and-thin matrix, $Y^T B$ essentially behaves like a sparse matrix-vector multiplication, which is a memory-bound operation. However, `r2-nmf-hier` is still much faster than all the other methods: On RCV1-full data set, `r2-nmf-hier`, `nmf-hier`, and `nmf-flat` cost about 7 minutes, 12.5 minutes, and 6.5 hours, respectively.

6.5 Clustering Quality

Clustering quality is evaluated on labeled data sets, shown in Fig. 4. The plot for the Cora data set is omitted for space reasons. For data sets with a defined hierarchy, ground-truth classes on the first 3 levels are used for evaluation, and those on deeper levels produce similar results. `nmf-hier` has identical results with `r2-nmf-hier`, thus is not shown here.

`r2-nmf-hier` is a very competitive method in general. NMF-based methods give stably good clustering quality using both the flat and hierarchical schemes. Compared to `nmf-flat`, we can clearly see the improved NMI values of

Table 2: Data sets used in our experiments.

Data sets	Labeled?	Has hierarchy?	Size	# terms	# docs	# nodes at each level
Reuters-21578	Y	N	medium-scale	12,411	7,984	20
20 Newsgroups	Y	Y	medium-scale	36,568	18,221	6/18/20
Cora	Y	N	large-scale	154,134	29,169	70
RCV1-labeled	Y	Y	large-scale	115,679	496,756	4/15/28/39/40
RCV1-full	N	-	large-scale	149,113	764,751	-

Figure 3: Timing results in seconds.

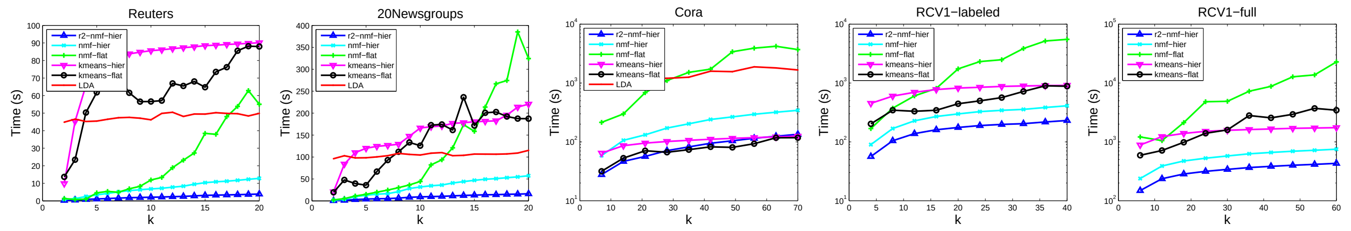


Figure 4: NMI on labeled data sets. Scales of y-axis for the same data set are set equal.

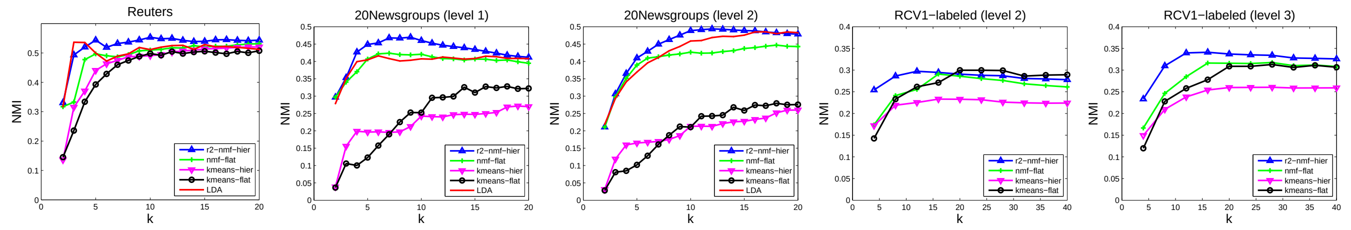
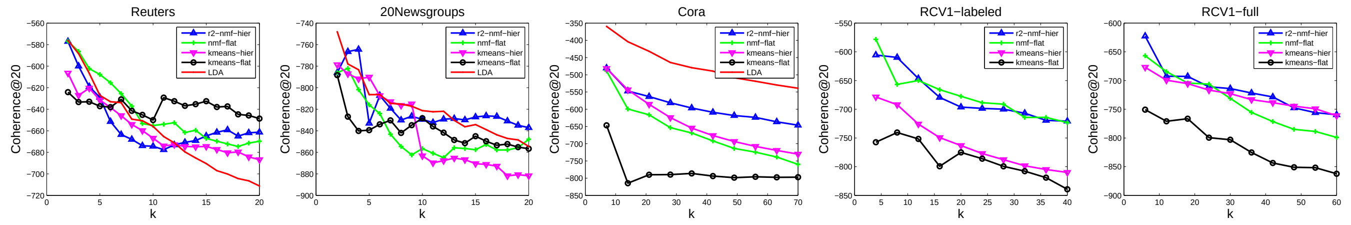


Figure 5: Coherence using the top 20 words for each topic.



`r2-nmf-hier`. Although `kmeans-flat` achieves comparable performances on RCV1-labeled, it performs poorly on other data sets. A general trend is that the improvement in clustering quality by `r2-nmf-hier` is more substantial when a deeper level of the defined hierarchy is used for evaluation, which correspond to more elaborated ground-truth classes.

We note that if NMI values are used for selecting the best number of clusters for a data set, `r2-nmf-hier` and `nmf-flat` frequently give different numbers (see the last two plots in Fig. 4). Thus they tend to interpret a data set in different ways. We also note that although `kmeans-hier` uses the same hierarchical clustering workflow as `r2-nmf-hier`, it performs poorly in most cases.

6.6 Semantic Quality of Topics

The coherence results for all the data sets are shown in Fig. 5. None of these methods have consistent performances when the number of clusters k is small; when k is large, `r2-nmf-hier` gives the highest coherence value in 3 out of 5 cases. On RCV1 data set, `r2-nmf-hier` is a stably good method in terms of topic coherence, while `nmf-flat` and `kmeans-hier` have comparable performances sometimes but perform very poorly otherwise. More study is needed to understand the benefits of each method in terms of topic coherence.

7. CONCLUSION

Hierarchical document clustering has a rich history in data analysis and management [23]. In this paper, we considered the divisive approach, which splits a data set in the top-down fashion and offers a global view of the data set compared to agglomerative clustering methods. In divisive hierarchical clustering, a clustering method is needed at each splitting step. However, it is not as easy as recursively applying any flat clustering method available to generate a tree structure. As can be seen in our experiments, the widely-used K-means clustering, when applied to hierarchical clustering, frequently generates very unbalanced clusters that lead to a poor organization of a corpus.

A good combination of a flat clustering method and a way to determine the next node to split is important for efficient and practical hierarchical clustering. In this paper, we proposed such a combination and showed its promising performance compared to other clustering methods such as NMF and LDA. For the efficiency of each splitting step, we designed a fast active-set-type algorithm for rank-2 NMF. Our algorithm has redundant computation but has continuous memory access, allowing better use of the cache; thus, it is faster than existing active-set-type algorithms. We also proposed a scoring method in the hierarchical clustering workflow, which provides a way to evaluate the potential of each leaf node to be split into two well-separated clusters and can be used to determine when to stop splitting. Outlier detection is also included in the overall workflow. Our method generated a binary tree structure of the full RCV1 data set in 7 minutes on a shared-memory machine with quad-core CPUs, compared to standard NMF which costs 6.5 hours.

We conclude by listing several shortcomings of the current method for further research. First, after a node is split, each document has a hard assignment to one of the two generated leaf nodes. It would be more flexible to enable soft assignments. Second, the performance of our proposed algorithm for rank-2 NMF is bounded by that of sparse matrix-vector multiplication (SpMV) when the data size is very large. The efficiency of our algorithm can be further boosted by using a more efficient SpMV implementation or moving to a distributed platform. Currently, our method can be used to build a hierarchical organization of documents efficiently on a single machine, possibly as part of a large machine learning infrastructure with many machines.

8. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation (NSF) grants CCF-0808863 and CCF-0732318.

9. REFERENCES

[1] S. Arora, R. Ge, Y. Halpern, D. M. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu. A practical algorithm for topic modeling with provable guarantees. http://arxiv.org/abs/1212.4777, 2012.

[2] D. M. Blei, T. L. Griffiths, M. I. Jordan, and J. B. Tenenbaum. Hierarchical topic models and the nested Chinese restaurant process. In *Advances in Neural Information Processing Systems 16*, 2003.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[4] J.-P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proc. Natl. Acad. Sci.*, 101(12):4164–4169, 2004.

[5] D. Cai, X. He, J. Han, and T. S. Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(8):1548–1560, 2011.

[6] C. Ding and X. He. Cluster merging and splitting in hierarchical clustering algorithms. In *ICDM '02: Proc. of the 2nd IEEE Int. Conf. on Data Mining*, pages 139–146, 2002.

[7] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters*, 26:127–136, 2000.

[8] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[9] H. Kim and H. Park. Sparse non-negative matrix factorizations via alternating non-negativity constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, 2007.

[10] H. Kim and H. Park. Nonnegative matrix factorization based on alternating non-negativity-constrained least squares and the active set method. *SIAM J. on Matrix Analysis and Applications*, 30(2):713–730, 2008.

[11] J. Kim and H. Park. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *Proc. of the Eighth IEEE Int. Conf. on Data Mining*, pages 353–362, 2008.

[12] D. Kuang, C. Ding, and H. Park. Symmetric nonnegative matrix factorization for graph clustering. In *SDM '12: Proc. of SIAM Int. Conf. on Data Mining*, pages 106–117, 2012.

[13] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems.* Prentice-Hall, Englewood Cliffs, NJ, 1974.

[14] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[15] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, 2004.

[16] T. Li, C. Ding, and M. I. Jordan. Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In *ICDM '07: Proc. of the 7th IEEE Int. Conf. on Data Mining*, pages 577–582, 2007.

[17] C.-J. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

[18] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, 2008.

[19] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of Internet portals with machine learning. *Inf. Retr.*, 3(2):127–163, 2000.

[20] D. Mimno, H. M. Wallach, E. Talley, M. Leenders, and A. McCallum. Optimizing semantic coherence in topic models. In *EMNLP '11: Proc. of the Conf. on Empirical Methods in Natural Language Processing*, pages 262–272, 2011.

[21] N. Sahoo, J. Callan, R. Krishnan, G. Duncan, and R. Padman. Incremental hierarchical clustering of text documents. In *CIKM '06: Proc. of the 15th ACM Int. Conf. on Information and Knowledge Management*, pages 262–272, 2011.

[22] M. H. Van Benthem and M. R. Keenan. Fast algorithm for the solution of large-scale non-negativity constrained least squares problems. *J. Chemometrics*, 18:441–450, 2004.

[23] P. Willett. Recent trends in hierarchic document clustering: a critical review. *Inf. Process. Manage.*, 24(5):577–597, 1988.

[24] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *SIGIR '03: Proc. of the 26th Int. ACM Conf. on Research and development in informaion retrieval*, pages 267–273, 2003.

[25] Y. Zhao, G. Karypis, and U. Fayyad. Hierarchical clustering algorithms for document datasets. *Data Min. Knowl. Discov.*, 10(2):141–168, 2005.