Abstract

The importance of unsupervised clustering and topic modeling is well recognized with ever-increasing volumes of text data. In this paper, we propose a fast method for hierarchical clustering and topic modeling called HierNMF2. Our method is based on fast Rank-2 nonnegative matrix factorization (NMF) that performs binary clustering and an efficient node splitting rule. Further utilizing the final leaf nodes generated in HierNMF2 and the idea of nonnegative least squares fitting, we propose a new clustering/topic modeling method called FlatNMF2 that recovers a flat clustering/topic modeling result in a very simple yet significantly more effective way than any other existing methods. We implement highly optimized open source software in C++ for both HierNMF2 and FlatNMF2 for hierarchical and partitional clustering/topic modeling of document data sets.

Substantial experimental tests are presented that illustrate significant improvements both in computational time as well as quality of solutions. We compare our methods to other clustering methods including K-means, standard NMF, and CLUTO, and also topic modeling methods including latent Dirichlet allocation (LDA) and recently proposed algorithms for NMF with separability constraints. Overall, we present efficient tools for analyzing large-scale data sets, and techniques that can be generalized to many other data analytics problem domains.

Keywords. Nonnegative matrix factorization, active-set methods, topic modeling, clustering.
1 Introduction

Enormous volumes of text data are generated daily due to rapid advances in text-based communication technologies such as smart phones, social media, and web-based text sources. Long articles in user-contributed encyclopedia and short text snippets such as tweets are two examples: The current English Wikipedia contains about 4.5 million articles¹ Twitter users worldwide generate over 400 million tweets every single day. Useful information can be extracted from these texts. For example, decision makers and researchers interested in the area of sustainability could learn from tweets how energy technology and policies receive public attention and affect daily lives. Analyzing the huge and increasing volume of text data efficiently has become an important data analytics problem.

We focus on unsupervised methods for analyzing text data in this paper. Many online texts have no label information; other documents such as Wikipedia articles are often tagged with multiple labels from a user-generated taxonomy thus do not fit into a traditional supervised learning framework well. Therefore, unsupervised clustering and topic modeling methods have become important tools for browsing and organizing a large text collection [5]. The goal of these unsupervised methods is to find a number of, say \( k \), document clusters where each cluster contains semantically connected documents and forms a coherent topic.

Topic modeling methods such as latent Dirichlet allocation (LDA) [6] are often based on probabilistic models. On the other hand, topics in document collections can be explained in a matrix approximation framework. Let \( \mathbb{R}_+ \) denote the set of nonnegative real numbers. In clustering and topic modeling, text data are commonly represented as a term-document matrix \( A \in \mathbb{R}_+^{m \times n} \). The \( m \) rows of \( A \) correspond to a vocabulary of \( m \) terms, and the \( n \) columns correspond to \( n \) documents. Consider the factorization of \( A \) as a product of two lower rank matrices:

\[
A = WH, \tag{1}
\]

where \( W \in \mathbb{R}_+^{m \times k} \) and \( H \in \mathbb{R}_+^{k \times n} \), and \( k \) is the number of topics we want to find, \( k << n \). Eq. (1) means that each of the \( n \) documents, say the \( i \)-th document \( a_i \), can be reconstructed by a linear combination of the \( k \) columns of \( W \), where the \( k \) linear coefficients are contained in the \( i \)-th column of \( H \). Suppose the given matrix \( A \) further satisfies \( \|a_i\|_1 = 1 \) (\( 1 \leq i \leq n \)), i.e., \( a_i \) represents the distribution over words in the \( i \)-th document. If we have normalized \( W \) so that each column of \( W \) sums to one, then each column of \( H \) would also sum to one. In this case, we can interpret the columns of \( W \) as distributions over words for the \( k \) topics and the columns of \( H \) as distributions over the \( k \) topics for the \( n \) documents. If we need to obtain a hard clustering result for the \( i \)-th document, we can select the topic corresponding to the largest entry in the \( i \)-th column of \( H \). This way, hard clustering, soft clustering, and topic modeling can be unified in the same model where \( W \) reveals the topics and \( H \) contains the cluster membership weights.

In reality, \( A = WH \) can only be approximately satisfied, that is,

\[
A \approx WH. \tag{2}
\]

Though (2) loses probabilistic interpretation of the generative process of topics and documents,¹

¹http://en.Wikipedia.org/Wiki/Wikipedia:Size_of_Wikipedia Retrieved in March 2014.
many recent studies have shown that it is also very useful for clustering and topic modeling \cite{2, 1, 21}. Note that $A, W, H$ are all nonnegative matrices. The formulation \cite{2} has been studied as nonnegative matrix factorization (NMF), a technique that approximates a nonnegative matrix by the product of two lower rank nonnegative matrices \cite{28, 23, 16}. The characteristics that distinguishes NMF from other matrix approximation methods is the nonnegativity constraints on $W$ and $H$, thus we can interpret these lower rank matrices more easily within the context of many application domains. NMF has been extensively used as a clustering method \cite{30, 7, 14, 20}, in which case the columns of $W$ are $k$ cluster representatives and the columns of $H$ are soft clustering assignments. In the case of text data, we typically normalize both the columns of $A$ and $W$ and scale the rows of $H$ accordingly. Because of the nonnegativity of $W$, a cluster representative can be interpreted as a vector of “importance” values for the $m$ terms and viewed as a topic; and because of the nonnegativity of $H$, a column of $H$ can be interpreted as topic mixture coefficients that approximately reconstruct the corresponding document though it does not sum to one. Therefore, in the framework of NMF, clustering and topic modeling can be viewed as two different names of the same analytics problem. Just as $W$ can be viewed as new basis vectors that represent the columns in the data matrix $A$, the topics which are the columns of $W$ provide us with a high level representation of the document collection $A$.

NMF as a topic modeling method has several advantages over LDA. First, without the need for a probabilistic interpretation, we can provide a term-document matrix with \textit{tf-idf} weighting as an input to NMF instead of raw frequencies of word occurrences, just as in most text classification methods \cite{26}. \textit{Tf-idf} weighting has been widely shown to improve classification and clustering accuracy. Second, numerous matrix computation and optimization routines that have been studied and implemented with high computational efficiency and rigorous analysis can provide a base to efficiently compute the solutions of NMF \cite{25, 15, 18, 8}, making NMF-based methods highly scalable for web-scale clustering and topic modeling.

Most of the previous work on clustering and topic modeling with NMF has been focused on a flat structure of topics. However, hierarchical clustering often reveals additional structures in the data. Recently, we proposed a very efficient NMF-based hierarchical clustering algorithm \cite{19}. This algorithm, which we call \textbf{HierNMF2}, recursively splits a data set into two clusters and generates a binary tree. The motivation for binary splitting was that we could exploit a special property of Rank-2 NMF, a.k.a. NMF with $k = 2$, and design an algorithm for computing Rank-2 NMF with high efficiency due to continuous memory access. To reveal the major topics and avoid splitting small clusters repeatedly, we also proposed a node splitting rule that adaptively determines the tree nodes to split.

In this paper, we will further accelerate HierNMF2 and propose a novel topic modeling method based on a hierarchy of clusters. The hierarchy generated by HierNMF2 does not fit well into a flat topic modeling \cite{4} or hierarchical topic modeling \cite{4} framework. We will introduce our way to flatten a hierarchy of clusters, called \textbf{FlatNMF2}, in order to produce flat clusters and topics that can be interpreted in a flat topic modeling framework. In short, we use the results obtained from HierNMF2 to obtain high quality estimates for the factor $W$ in NMF \cite{2}. The resulting FlatNMF2 is orders of magnitude faster and more effective than existing topic modeling methods such as the standard NMF and LDA: the cluster/topic quality of its solutions often retains or even exceeds that of the standard NMF and LDA. We implement HierNMF2 and FlatNMF2 in highly optimized
software written in C++, with a custom implementation of the routine for the multiplication of a large sparse matrix with a tall-skinny dense matrix (SpMM).

We summarize our contribution in this paper as follows:

1. By combining Rank-2 NMF with a node splitting rule, we develop HierNMF2, an efficient workflow for hierarchical document clustering. Our methodology is able to determine both the tree structure and the depth of the tree on-the-fly and detect outliers, in contrast to hierarchical probabilistic modeling methods [4] that require the depth of the tree be specified by the user.

2. Based on the idea of hierarchical clustering, we propose FlatNMF2 by transforming the tree structure into a flat structure of clusters. This is a novel topic modeling method more efficient and effective than previous Expectation Maximization (EM)-based methods for large-scale text data.

3. We describe the challenges to building effective software for analyzing large volumes of text using hierarchical clustering and topic modeling, and our solutions to these challenges. The key contribution is the accelerated routine for SpMM on multi-core CPUs. We expect that our work has a large impact on the efficiency of many data analytics problems as the sizes of sparse data matrices are steadily growing. We implement HierNMF2 and FlatNMF2 with accelerated SpMM as highly optimized open source software in C++.

The rest of this paper is organized as follows. In Section 2 we conduct detailed analysis of existing active-set-type algorithms for NMF in the special case of $k = 2$. In Section 3 we present our algorithm for solving Rank-2 NMF. In Section 4 we describe the node splitting rule and the workflow of HierNMF2. In Section 5 we propose a method to flatten a hierarchy of clusters, i.e., FlatNMF2 that computes a Rank-$k$ NMF efficiently. In Section 6 we describe the efficiency challenges in HierNMF2/FlatNMF2 and the accelerated routine for SpMM. In Section 7 we present the performance results of our HierNMF2 and FlatNMF2 software on large-scale text data. In Section 8 we provide conclusions and discuss future directions.

The Rank-2 NMF algorithm and the hierarchical clustering workflow were first proposed in our previous work [19], which are included in this paper for completeness. The new contributions in this paper include an approach to transform the discovered hierarchy to a flat partitioning with improved cluster/topic quality, open source C++ software for HierNMF2/FlatNMF2 with accelerated SpMM routines, and substantial experimental results on web-scale text data sets.

2 Alternating Nonnegative Least Squares (ANLS) for NMF

In this paper, we consider algorithms for NMF that fit into a two-block coordinate descent framework [25, 15, 18, 16] due to their superior theoretical convergence properties. To find $W$ and $H$ whose product well approximates a nonnegative matrix $A$ as in (2), it is a common practice to form a constrained optimization problem that minimizes the difference between $A$ and $WH$ [28, 23, 16]:

$$\min_{W,H \geq 0} \| A - WH \|_F^2,$$ (3)
where \( \| \cdot \| \) denotes the Frobenius norm. The formulation (3) is nonconvex in all the variables in \( W \) and \( H \), whereas it is convex in \( W \) only and convex in \( H \) only. In the two-block coordinate descent framework, we divide the variables into two blocks corresponding to \( W \) and \( H \), and iteratively solve \( W \) and \( H \) by alternating between two subproblems until some stopping criterion is satisfied:

\[
\begin{align*}
\min_{W \geq 0} & \| H^T W^T - A^T \|_F^2, \\
\min_{H \geq 0} & \| W H - A \|_F^2.
\end{align*}
\] (4a) (4b)

When an optimal solution is obtained for each subproblem (4) in each iteration, this iterative procedure is guaranteed to converge to a stationary point [12], which is a good convergence property for nonconvex problems. Both of the two subproblems (4) are nonnegative least squares (NLS) problems [22] with multiple right-hand sides. The two-block coordinate descent algorithms for NMF differ essentially in the way these NLS problems are solved. In the rest of the paper, we use the following notation to denote NLS:

\[
\begin{align*}
\min_{G \geq 0} & \| B G - Y \|_F^2,
\end{align*}
\] (5)

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 found.

Various algorithms can be used to solve the NLS problem and can be categorized into standard optimization algorithms and active-set-type algorithms. A classical active-set algorithm for NLS with a single right-hand side was introduced in [22]. In the context of NMF, Lin [25] claimed that it would be expensive to solve NLS with multiple right-hand sides using the active-set algorithm repeatedly, and proposed a projected gradient descent (PGD) algorithm. However, Kim & Park [15] proposed several improvements for the original active-set algorithm, and achieved an NMF algorithm with overall efficiency comparable to PGD. Later, a block principle pivoting algorithm for NLS [18] was proposed, which proved to be more efficient than the active-set algorithm when \( k \) is large. We call both active-set and block pivoting type algorithms for NMF as active-set-type algorithms.

In active-set-type algorithms for NLS, 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. It should be pointed out that using an active-set-type method to generate this least squares problem is robust to rank deficiency since the columns of the coefficient matrix corresponding to the passive set indices are linearly independent at each iteration and the solution is feasible [10]. 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 principle pivoting methods. To improve the efficiency of solving NLS with multiple right-hand sides [5], the columns of \( G \) with the same active set pattern are grouped together for lower computational complexity and more cache-efficient computation [13 29], and the grouping of columns changes when the active set is re-identified in each iteration of NLS. 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, as in hierarchical clustering that
generates a binary tree, the number of possible active sets is reduced to $2^2 = 4$ for each column of $G$, thus it is practical to enumerate all active sets. Conceptually, active-set-type algorithms search for the optimal active set in a finite number 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 iterations for 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 NLS, 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, using the active-set based algorithm [15] as an exemplar algorithm. Given an $m \times n$ sparse data matrix $A$ and the number of clusters $k$, the complexity of one NLS run for standard NMF is upper bounded by:

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

where $N$ is the number of nonzero entries in $X$, $t$ is the number of search steps towards the optimal active set, and flops denotes floating point operations.

In hierarchical clustering, we need to perform Rank-2 NMF $k - 1$ times, and the complexity of one NLS run summed over all the $k - 1$ steps is at most:

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

The actual flops in hierarchical clustering must be smaller than (7), because any splitting other than the first step is executed on a subset of the data set only. Thus, the expression (6) is superlinear with respect to $k$, while (7) 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 for Rank-2 NMF in this paper, the efficiency of NMF-based hierarchical clustering will be boosted even more.

### 3 A Fast Algorithm for Nonnegative Least Squares with Two Basis Vectors

In ANLS, the problem of solving NMF is reduced to the problem of solving NLS with multiple right-hand sides: $\min_{G \geq 0} \|BG-Y\|^2_F$. Now we focus on solving NLS where $B$ has two columns. In the context of NMF, $Y$ is set to be either the data matrix $X$, or $X^T$. Let $Y = [y_1, \cdots, y_n], G = [g_1, \cdots, g_n]$. We emphasize that $Y \geq 0$ in the NLS problem to be solved since the data is nonnegative, and $B \geq 0$ since $B$ represents the nonnegative factor $W$ or $H^T$.

---

\footnote{In (6), the terms $4kN$ and $2(m + n)k^2$ come from forming the left and right hand sides of the normal equation in one NLS run, and the terms $t \cdot (1/3)k^3$ and $t \cdot 2(m + n)k^2$ come from Cholesky factorizations and triangular solves in solving unconstrained least squares problems for the inner iterations in one NLS run.}
Table 1: Four possible active sets when $B \in \mathbb{R}^{m \times 2}$.

| $\mathcal{A}$ | $\mathcal{P}$ | $J(g)$ |
|---------------|---------------|---------|
| $\{1,2\}$    | $\emptyset$  | $\|y\|_2^2$ |
| $\{1\}$      | $\{2\}$     | $\|b_{2g_2} - y\|_2^2$ |
| $\{2\}$      | $\{1\}$     | $\|b_{1g_1} - y\|_2^2$ |
| $\emptyset$   | $\{1,2\}$   | $\|b_{1g_1} + b_{2g_2} - y\|_2^2$ |

Since the formulation (5) for NLS with multiple right-hand sides can be rewritten as

$$
\min_{g_1, \ldots, g_n \geq 0} \|Bg_1 - y_1\|_2^2 + \cdots + \|Bg_n - y_n\|_2^2,
$$

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

### 3.1 Efficient Solution of $\min_{g \geq 0} \|Bg - y\|_2$

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 $g$ that correspond to the active set and the passive set as $g_\mathcal{A}$ and $g_\mathcal{P}$, respectively. At each iteration of the algorithm, $g_\mathcal{A}$ is set to zero, and $g_\mathcal{P}$ is solved by unconstrained least squares using a subset of columns of $B$ corresponding to $\mathcal{P}$. The optimal passive set is the one where the solution of unconstrained least squares is feasible [10], i.e., $g_\mathcal{P} > 0$, and $\|Bg - y\|_2^2$ is minimized.

When $k = 2$, we have

$$
J(g) \equiv \|Bg - y\|_2^2 = \|b_{1g_1} + b_{2g_2} - y\|_2^2,
$$

where $B = [b_1, b_2] \in \mathbb{R}^{m \times 2}$, $y \in \mathbb{R}^{m \times 1}$, and $g = [g_1, g_2]^T \in \mathbb{R}^{2 \times 1}$. 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 $J(g)$ in Table 1 by solving the unconstrained least squares problem. Then, of all the feasible solutions of $g$ (i.e. $g \geq 0$), we pick the one with the smallest $J(g)$. 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.

First, we claim that the two unconstrained problems $\min \|b_{1g_1} - y\|_2$, and $\min \|b_{2g_2} - y\|_2$ always yield feasible solutions. Take $\min \|b_{1g_1} - y\|_2^2$ as an example. Its solution is:

$$
g_1^* = \frac{y^T b_1}{b_1^T b_1}.
$$

Geometrically, the best approximation of vector $y$ in the one-dimensional space spanned by $b_1$ is the orthogonal projection of $y$ onto $b_1$. If $b_1 \neq 0$, we always have $g_1^* \geq 0$ since $y \geq 0$, $b_1 \geq 0$. In the context of Rank-2 NMF, the columns of $W$ and the rows of $H$ are usually linearly independent.
Algorithm 1 Algorithm for solving $\min_{\mathbf{g} \geq 0} \| \mathbf{B} \mathbf{g} - \mathbf{y} \|_2^2$, where $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2] \in \mathbb{R}_+^{m \times 2}, \mathbf{y} \in \mathbb{R}_+^{m \times 1}$

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

when nonnegative-rank($X$) $\geq 2$, thus $\mathbf{b}_1 \neq 0$ holds in practice.

If $\mathbf{g}^0 \equiv \arg \min \| \mathbf{b}_1 \mathbf{g}_1 + \mathbf{b}_2 \mathbf{g}_2 - \mathbf{y} \|_2^2$ is nonnegative, then $\mathcal{A} = \emptyset$ is the optimal active set because the unconstrained solution $\mathbf{g}^0$ is feasible and neither $\min \| \mathbf{b}_1 \mathbf{g}_1 - \mathbf{y} \|_2^2$ nor $\min \| \mathbf{b}_2 \mathbf{g}_2 - \mathbf{y} \|_2^2$ can be smaller than $\mathcal{J}(\mathbf{g}^0)$. Otherwise, we only need to find the smallest objective $\mathcal{J}(\mathbf{g})$ among the other three cases since they all yield feasible solutions. However, $\mathcal{A} = \{1, 2\}$, i.e., $\mathcal{P} = \emptyset$, can be excluded, since there is always a better solution than $\mathbf{g} = (0,0)^T$. Using $g_1^*$, the solution of $\min \| \mathbf{b}_1 \mathbf{g}_1 - \mathbf{y} \|_2^2$, we have

$$\| \mathbf{b}_1 g_1^* - \mathbf{y} \|_2^2 = \| \mathbf{y} \|_2^2 - (\mathbf{y}^T \mathbf{b}_1)^2/(\mathbf{b}_1^T \mathbf{b}_1) \leq \| \mathbf{y} \|_2^2.$$  \hspace{1em} (11)

To compare $\| \mathbf{b}_1 g_1^* - \mathbf{y} \|_2^2$ and $\| \mathbf{b}_2 g_2^* - \mathbf{y} \|_2^2$, we note that $(\mathbf{b}_1 g_1^* - \mathbf{y}) \perp \mathbf{b}_1 g_1^*$ and $(\mathbf{b}_2 g_2^* - \mathbf{y}) \perp \mathbf{b}_2 g_2^*$, therefore we have

$$\| \mathbf{b}_j g_j^* \|_2^2 + \| \mathbf{b}_j g_j^* - \mathbf{y} \|_2^2 = \| \mathbf{y} \|_2^2$$  \hspace{1em} (12)

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

Our algorithm for NLS with a single right-hand side is summarized in Algorithm 1. Note that $\mathbf{B}^T \mathbf{B}$ and $\mathbf{B}^T \mathbf{y}$ need to be computed only once for lines 1, 5, 6.

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

When Algorithm 1 is applied to NLS with multiple right-hand sides, computing $\mathbf{g}^0, g_1^*, g_2^*$ for each vector $\mathbf{y}_i$ separately is not cache-efficient. In Algorithm 2 we solve NLS with $n$ different vectors $\mathbf{y}_i$ simultaneously, and the analysis in Section 3.1 becomes important. Note that the entire for-loop (lines 5-15, Algorithm 2) is embarrassingly parallel and can be vectorized. To achieve this, unconstrained solutions for all three possible passive 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 $\mathbf{g}_i^0 \geq 0$ (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.

Note that a naïve implementation of comparing $\| \mathbf{b}_1 g_1^* - \mathbf{y} \|_2^2$ and $\| \mathbf{b}_2 g_2^* - \mathbf{y} \|_2^2$ for $n$ different vectors $\mathbf{y}$ requires $O(mn)$ complexity due to the creation of the $m \times n$ dense matrix $\mathbf{B} \mathbf{G} - \mathbf{Y}$.  

8
Algorithm 2 Algorithm for solving $\min_{G \geq 0} \|BG - Y\|_F^2$, where $B = [b_1, b_2] \in \mathbb{R}_+^{m \times 2}$, $Y \in \mathbb{R}_+^{m \times n}$

1: Solve unconstrained least squares $G^0 = [g^0_1, \cdots, g^0_n] \leftarrow \min \|BG - Y\|_F^2$ by normal equation $B^T B G = B^T Y$
2: $\beta_1 \leftarrow \|b_1\|$, $\beta_2 \leftarrow \|b_2\|$  
3: $u \leftarrow (Y^T b_1)/\beta_1^2$  
4: $v \leftarrow (Y^T b_2)/\beta_2^2$  
5: for $i = 1$ to $n$ do  
6: if $g^0_i \geq 0$ then  
7: return $g^0_i$  
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

In contrast, our algorithm only requires $O(m + n)$ complexity at this step (line 9, Algorithm 2), because $b_1, b_2$ are the same across all the $n$ right-hand sides. Assuming $Y$ is a sparse matrix with $N$ nonzeros, the overall complexity of Algorithm 2 is $O(N)$, which is the same as the complexity of existing active-set-type algorithms when $k = 2$ (see Eq. 6). The dominant part comes from computing the matrix product $Y^T B$ in unconstrained least squares.

4 HierNMF2

Rank-2 NMF can be recursively applied to a data set, generating a hierarchical tree structure. In this section, we focus on text analytics and develop an overall efficient approach to hierarchical document clustering, which we call HierNMF2. When constructing the tree in the HierNMF2 workflow, we need to: (1) choose an existing leaf node at each splitting step and generate two newer leaf nodes; (2) determine tiny clusters as outlier documents that do not form a major theme in the data set; (3) determine when a leaf node should not be split and should be treated as a “permanent leaf node”. In the following, we focus mainly on the node splitting rule.

Extensive criteria for selecting the next leaf node to split were discussed in previous literature for general clustering methods [2], 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 [30], 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 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 to be computed only once when the node first appears in the tree. For an illustration of a leaf node and its two potential children, see Fig. 1. We split a leaf node $\mathcal{N}$ if at least two well-separated topics can be discovered
within the node. Thus we expect that \( N \) receives a high score if the top words for \( N \) are a well-balanced combination of the top words for its two potential children, \( \mathcal{L} \) and \( \mathcal{R} \). We also expect that \( N \) receives a low score if the top words for \( \mathcal{L} \) and \( \mathcal{R} \) are almost the same.

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

\[ f_1, f_2, \ldots, f_m, \]

and the shuffled orderings in \( f_L \) and \( f_R \) are respectively:

\[ f_{i_1}, f_{i_2}, \ldots, f_{i_m}; \]
\[ f_{r_1}, f_{r_2}, \ldots, f_{r_m}. \]

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

\[
p(f_i) = \log (m - \max\{i_1, i_2\} + 1),
\]

\[
g(f_i) = \frac{\log (m - i + 1)}{p(f_i)},
\]

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 \( f_S \) (\( f_S = f_L \) or \( f_R \)), mNDCG is defined as:

\[
m\text{DCG}(f_S) = g(f_{s_1}) + \sum_{i=2}^{m} \frac{g(f_{s_i})}{\log_2(i)},
\]

\[
m\text{IDCG} = \hat{g}_1 + \sum_{i=2}^{m} \frac{\hat{g}_i}{\log_2(i)},
\]

\[
m\text{NDCG}(f_S) = \frac{m\text{DCG}(f_S)}{m\text{IDCG}}.
\]
As we can see, mNDCG is computed basically 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:

\[
\text{score}(\mathcal{N}) = \text{mNDCG}(f_L) \times \text{mNDCG}(f_R).
\]  

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

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

2. When both \( L \) and \( R \) 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 incur a large discount, and both mNDCG\( (f_L) \) and mNDCG\( (f_R) \) will be small.

3. When \( L \) describes the same topic as that of \( \mathcal{N} \), and \( R \) describes a totally unrelated topic (e.g. outliers in \( \mathcal{N} \)), then mNDCG\( (f_L) \) is large and mNDCG\( (f_R) \) 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 interchangeably. The while-loop in this workflow (lines 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 conclude that one (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. Empirically, without the outlier detection procedure, the constructed tree would end up with many tiny leaf nodes, which do not correspond to salient topics and would degrade the clustering quality. We have not specified the best moment to exit and stop the recursive splitting process. Our approach is to simply set an upper limit on the number of leaf nodes \( k \). However, other strategies can be used to determine when to exit, such as specifying a score threshold \( \sigma \) and exiting the program when none of the leaf nodes have scores above \( \sigma \); \( \sigma = 0 \) 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 incurring \( O(n^2) \) overhead [9], 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 performance using clustering quality measures in the Experiments section.

5 FlatNMF2

Although hierarchical clustering often provides a more detailed taxonomy than flat clustering, a tree structure of clusters cannot be interpreted in any existing probabilistic topic modeling framework [6, 4]. Often flat clusters and topics are also required for visualization purposes. Therefore, we
Algorithm 3 HierNMF2: 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 $R$, containing all the $n$ documents
3: $\text{score}(R) \leftarrow \infty$
4: repeat
5: $M \leftarrow$ 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 $M$ and create two potential children $N_1, N_2$, where $|N_1| \geq |N_2|$
10: if $|N_1| \geq \beta |N_2|$ and $\text{score}(N_2)$ is smaller than every positive score of current leaf nodes then
11: $Z \leftarrow Z \cup N_2$, $M \leftarrow M - Z$, $i \leftarrow i + 1$
12: else
13: break
14: end if
15: end while
16: if $i < T$ then
17: Split $M$ into $N_1$ and $N_2$ (hard clustering)
18: Compute $\text{score}(N_1)$ and $\text{score}(N_2)$
19: else
20: $M \leftarrow M \cup Z$ (recycle the outliers and do not split $M$)
21: $\text{score}(M) \leftarrow -1$ (set $M$ 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

We present a method to recover flat clusters/topics from the HierNMF2 result. We call our algorithm FlatNMF2, a new method for large-scale topic modeling.

We formulate the problem of flattening a hierarchy of clusters as an NLS problem. Assume that at the end of HierNMF2, we obtain a hierarchy with $k$ leaf nodes. Each tree node $N$ is associated with a multinomial term distribution represented as a vector of length $m$. We treat each vector associated with a leaf node as a topic and collect all these vectors, forming a term-topic matrix $\hat{W} \in \mathbb{R}^{m \times k}$. This matrix can be seen as a topic model after each column is normalized. We compute an approximation of the term-document matrix $A$ using $\hat{W}$:

$$\min_{H \geq 0} \| \hat{W} H - A \|^2_F.$$ \hspace{1cm} (19)

This is an NLS problem with $k$ basis vectors and can be solved by many existing algorithms \cite{15,18,16}.

Now we describe two intuitive and efficient ways to determine the topic vector at each leaf node. First, we can form the vector $w_N$ at a leaf node $N$ as one of the two columns of $W$ given by the NMF result that generates the node $N$ along with its sibling, which does not require further computation. Second, we can use the leading left singular vector from the singular value decomposition (SVD)
of the data submatrix at the node \( \mathcal{N} \) as the representative of \( \mathcal{N} \), which requires calling a sparse eigensolver to compute the SVD. Empirically, we found that the former gives better clustering accuracy and topic quality and is also more efficient, and thus we form the matrix \( \hat{W} \) in this way – directly using the results from Rank-2 NMF – and report the experimental results of FlatNMF2 in Section 7.

Just as in the original NMF, the matrix \( H \) in the solution of (19) can be treated as soft clustering assignments, and we can obtain a hard clustering assignment for the \( i \)-th document by selecting the index associated with the largest element in the \( i \)-th column of \( H \).

### 6 SpMM in HierNMF2/FlatNMF2

A major obstacle to achieving lightning fast performance of HierNMF2 and FlatNMF2 is the multiplication of a large sparse matrix with a tall-skinny dense matrix (SpMM). Most existing algorithms for solving NLS as well as Algorithm 2 we proposed for solving NLS with two basis vectors include a matrix multiplication step, that is, \( AH^T \) in (4a) and \( A^T W \) in (4b). When \( A \) is a sparse matrix, such as the term-document matrix in text analysis, this step calls the SpMM routine since \( k << \min(m, n) \). In the special case of Rank-2 NMF, the dense matrix contains two columns. Fig. 2 shows the timing of various steps of the entire workflow of HierNMF2 on the 800K RCV1 and 4.5M Wikipedia articles data sets (see Table 2). For HierNMF2, SpMM costs 67% and 86% of the total runtime for these two data sets respectively, and thus is the major bottleneck.

We implement our custom routines of SpMM by exploiting multiple threads on multi-core CPUs and placing our focus on the cases with a few (up to 16) dense columns. We use a simple strategy where one thread is responsible for the multiplication of the sparse matrix and a single column of the dense matrix. Though this strategy requires accessing the sparse matrix in memory for multiple times, we have found that for \( k \leq 256 \), our routine outperforms both Matlab and Intel Math Kernel Library that are restricted to using one CPU core for sparse matrix operations. A benchmarking result with varying \( k \) is shown in Fig. 3.

SpMM is a common bottleneck in many other machine learning algorithms. In data analytics where a sparse data matrix is involved, e.g. term-document matrices, user-by-item rating matrices, and graph adjacency matrices, the matrix often has an irregular sparsity structure. Such sparse matrices are commonly stored in a generic format such as the Compressed Sparse Row format that restricts the use of efficient SpMM routines for matrices with special sparsity structures, such as those with small dense blocks in finite element methods. Therefore, our software would greatly benefit applications beyond document clustering and topic modeling.
Figure 2: Timing of the major algorithmic steps in NMF-based hierarchical clustering shown in different colors. The legends are: \textbf{SpMM} – Sparse-dense matrix multiplication, where the dense matrix has two columns; \textbf{memcpy} – Memory copy for extracting a submatrix of the term-document matrix for each node in the hierarchy; \textbf{opt-act} – Searching for the optimal active set in active-set-type algorithms \cite{15, 18, 16}; \textbf{misc} – Other algorithmic steps altogether. “Previous NMF algorithms” refer to active-set based algorithms \cite{15, 17, 18}. The Rank-2 NMF algorithm greatly reduced the cost of \textit{opt-act}, leaving SpMM as the major bottleneck.

7 Experiments

In this section, we describe our experimental settings and demonstrate both the efficiency and quality of HierNMF2 and FlatNMF2. The small- to medium-scale experiments were run on a platform with two Intel Xeon X5550 quad-core processors and 24 GB memory, and the large-scale experiments were run on a server with two Intel E5-2620 processors, each having six cores, and 377 GB memory.
Figure 3: Run-time of the multiplication of an $N \times N$ sparse matrix and a $N \times K$ dense matrix (SpMM) with varying $K$, using the SpMM routines in Matlab, Intel Math Kernel Library (MKL), and our SmallK software library, with double precision. The density (fraction of nonzeros) of the sparse matrix is $10^{-4}$. The benchmarking results are collected on a machine with two 8-core Intel Xeon E5-2670 processors and 64 GB memory. SmallK uses 16 threads in the benchmarking.

7.1 Data Sets

Six text data sets were used in our experiments: 1. Reuters-21578 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 contains articles from Usenet newsgroups and has a defined hierarchy of 3 levels. Usenet users post messages and reply to posts under various discussion boards, often including a personalized signature at the end of their messages. Unlike the widely-used indexing of this data set, we observed that many articles had duplicate paragraphs due to cross-referencing. We discarded cited paragraphs and signatures, which posed a more difficult clustering problem. 3. Cora is a collection of research papers in computer science, from which we extracted the title, abstract, and reference-contexts. Although this data set comes with a predefined topic hierarchy of 3 levels, we observed that some topics, such as “AI – NLP” and “IR – Extraction”, were closely related but resided in different subtrees. Thus, we ignored the hierarchy and obtained 70 ground-truth classes as a flat partitioning. 4. NIPS is a collection of NIPS conference papers. We chose the 447 papers in the 2001-2003 period, which were associated with labels indicating the technical area (algorithms, learning theory, vision science, etc). 5. RCV1 is a much larger collection of news articles from Reuters, containing about 800,000 articles from the time period of 1996-1997. We used the entire collection as an unlabeled data set. 6. Wikipedia is an online, user-contributed encyclopedia and provides periodic dumps of the entire website. We processed the dump of all the English Wikipedia articles from March 2014, and used the resulting 4.5 million documents as an unlabeled data set Wiki-4.5M, ignoring user-defined categories.

We summarize these data sets in Table 3. The first four data sets are medium-scale and have...
Table 3: Data sets used in our experiments. Numbers in parentheses are the numbers of clusters/topics we requested for unlabeled data sets.

| Data sets          | Has label | Has hierarchy | # terms | # docs | # nodes at each level |
|--------------------|-----------|---------------|---------|--------|-----------------------|
| Reuters-21578      | Y         | N             | 12,411  | 7,984  | 20                    |
| 20 Newsgroups      | Y         | Y             | 36,568  | 18,221 | 6/18/20               |
| Cora               | Y         | N             | 154,134 | 29,169 | 70                    |
| NIPS               | Y         | N             | 17,981  | 447    | 13                    |
| RCV1               | N         | -             | 149,113 | 764,751| (60)                  |
| Wiki-4.5M          | N         | -             | 2,361,566 | 4,126,013 | (80)           |

ground-truth labels for the evaluation of cluster quality, while the remaining two are large scale and treated as unlabeled data sets. All the labeled data sets except 20 Newsgroups have very unbalanced sizes of ground-truth classes. We constructed the normalized-cut weighted version of term-document matrices as in [30].

7.2 Implementation

We implemented HierNMF2 and FlatNMF2 both in Matlab and in an open-source C++ software library called smallk\(^6\). The methods we compared are grouped into two categories: clustering methods and topic modeling methods. Though clustering and topic modeling can be unified in the framework of matrix factorization, as explained in Section 1, we label a method as belonging to one of the two categories according to the task it was originally targeted for.

The clustering methods we compared include:

- **nmf-hier**: Hierarchical clustering based on standard NMF with ANLS and an active-set method for NLS [15].
  The active-set method searches through the space of active-set/passive-set partitionings for the optimal active set, with a strategy that reduces the objective function at each search step.

- **nmf-flat**: Flat clustering based on standard NMF with ANLS and block principal pivoting method for NLS [17].
  In our experiments, multiplicative update rule algorithms [23] for standard NMF were always slower and gave similar quality compared to active-set-type algorithms, thus were not included in our results.

- **kmeans-hier**: Hierarchical clustering based on standard K-means.
  We used the hierarchical clustering workflow described in Algorithm 3 however, the term distribution associated with each node was given by the centroid vector from the K-means run that generates this node.

- **kmeans-flat**: Flat clustering based on standard K-means.

[^6]: https://smallk.github.io/
• **CLUTO**: A clustering toolkit written in C++. We used the default method in its *vluster* program, namely a repeated bisection algorithm.

The topic modeling methods we compared include:

• **Mallet-LDA**: The software MALLET written in Java for flat topic modeling, which uses the Gibbs sampling algorithm for LDA. 1000 iterations were used by default.

• **AnchorRecovery**: A recent fast algorithm to solve NMF with separability constraints. It selects an “anchor” word for each topic, for example, “Los Angeles Clippers” rather than “basketball”, which could carry a narrow meaning and not semantically represent the topic. The software is written in Java. We used the default parameters.

• **XRAY**: Another recent algorithm to solve NMF with separability constraints. It incrementally selects “extreme rays” to find a cone that contains all the data points. We used the *greedy* option as the selection criteria in this algorithm.

• **Hottopixx**: A recent method that formulated Separable NMF as a linear program and solved it using incremental gradient descent. We used the default parameters.

We use the normalized mutual information (NMI) measure to evaluate the cluster and topic quality. This is a measure of the similarity between two flat partitionings, and is only applicable to data sets for which the ground-truth labels are known. It is particularly useful when the number of generated clusters is different from that of ground-truth labels and can be used to determine the optimal number of clusters. More details can be found in [26]. For data sets with defined hierarchy, we compute NMI between a generated partitioning and the ground-truth classes at each level of the ground-truth tree. In other words, if the ground-truth tree has depth $L$, we compute $L$ NMI measures corresponding to each level. When evaluating the results given by HierNMF2 (Algorithm 3), we treat all the outliers as one separate cluster for fair evaluation.

Hierarchical clusters and flat clusters cannot be compared against each other directly. When evaluating the hierarchical clusters given by HierNMF2, we take snapshots of the tree as leaf nodes are generated, and treat all the leaf nodes in each snapshot as a flat partitioning which is to be compared against the ground-truth classes. This is possible since the leaf nodes are non-overlapping. Thus, if the maximum number of leaf nodes is set to $c$, we produce $c-1$ flat partitionings forming a hierarchy. For each method, we perform 20 runs 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, · · · , $c$.

The maximum number of leaf nodes $c$ is set to be the number of ground-truth labels at the deepest level for labeled data sets (see Table 3), and we set $c = 60$ for RCV1 and $c = 80$ for Wiki4.5M. The hierarchical clustering workflow (Algorithm 3) runs with parameters $\beta = 9$, $T = 3$. The Matlab *kmeans* function has a batch-update phase and a more time-consuming online-update phase. We rewrote this function using BLAS-3 operations and boosted its efficiency substantially.

---

7 [http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview](http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview)
8 [http://mallet.cs.umass.edu/](http://mallet.cs.umass.edu/)
9 [https://github.com/mimno/anchor](https://github.com/mimno/anchor)
10 [http://www.cc.gatech.edu/~dkuang3/software/kmeans3.html](http://www.cc.gatech.edu/~dkuang3/software/kmeans3.html)
7.3 Cluster Quality

Figs. 4 and 5 show the cluster quality on four labeled data sets, comparing HierNMF2/FlatNMF2 with the state-of-the-art clustering methods and topic modeling methods, respectively. nmf-hier generates the identical results with HierNMF2 (but the former is less efficient) and is not shown in Fig. 4.

We can see that HierNMF2 is a very competitive method in general, achieving comparable NMI.
values with those of previous algorithms. FlatNMF2 gives consistently better cluster and topic quality in almost every case. Especially on the Cora data set where HierNMF2 performs worse than NMF and LDA, the flattened clusters/topics in FlatNMF2 achieved the best NMI values. One possible reason for the better performance of FlatNMF2 is that documents that appear to be outliers are removed when building the hierarchy in HierNMF2, and thus the topics at the leaf nodes are more meaningful and represent more salient topics than those generated by a flat topic modeling method that takes every document into account. The algorithms solving NMF with separability constraints yielded the lowest clustering quality. Among them, AnchorRecovery and Hottopixx both require several parameters provided by the user, which could be time-consuming to tune and have a large impact on the performance of their algorithms. We used the default parameters for both of these methods, which may have negatively affected their NMIs.
Figure 6: Timing results for the Matlab implementation of HierNMF2, FlatNMF2, NMF, and K-means on the smaller data sets.

7.4 Timing Results

Fig. 6 shows the run-time of the proposed methods versus NMF and K-means, all implemented in Matlab. HierNMF2 and FlatNMF2 required substantially less run-time compared to the standard flat NMF. These results 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. For example, to generate 70 clusters on the Cora data set, HierNMF2, FlatNMF2, nmf-hier, and nmf-flat took about 2.4, 2.6, 5.6, and 55.3 minutes, respectively. We note that K-means with only the batch-update phase has similar run-time to HierNMF2 and FlatNMF2; however, their cluster quality is not as good, which was shown earlier in Fig. 4.

Fig. 7 compares the run-time of our C++ implementation of HierNMF2/FlatNMF2 available in
the software smallk versus off-the-shelf toolkits (CLUTO, Mallet-LDA) and recent methods proposed for large-scale topic modeling, namely AnchorRecovery, XRAY, and Hottopixx. We used 8 threads when possible to set the number of threads manually (in the cases of smallk, CLUTO, Mallet-LDA, and Hottopixx).

On the RCV1 and Wiki-4.5M data sets, HierNMF2 and FlatNMF2 are about 20 times faster than Mallet-LDA; particularly on the largest Wiki-4.5M data set in our experiments, HierNMF2 found 80 topics in about 50 minutes, greatly enhancing the practicality of topic modeling algorithms when compared to the other software packages in our experiments.

The three algorithms AnchorRecovery, XRAY, and Hottopixx that solve NMF with separability constraints require a large $m \times m$ matrix, i.e. word-word similarities. We reduced the vocabulary of Wiki-4.5M to about 100,000 unique terms in order to accommodate the $m \times m$ matrix in main memory for these algorithms. Among them, XRAY and Hottopixx build a dense word-word
Figure 8: Hierarchical clustering result generated on a small data set – 954 customer reviews on Edmunds.com – for illustration. The hierarchy is automatically detected and not necessarily a balanced tree. Each tree node $N$ is associated with a column of $W$, denoted as $w_N$, generated by Rank-2 NMF applied on its parent node. We display the five terms with highest importance values in $w_N$. Red boxes indicate leaf nodes while blue boxes indicate non-leaf nodes. The number in the parentheses at each node indicates the number of documents associated with that node.

similarity matrix and thus have a large memory footprint [21, 3]. AnchorRecovery, on the other hand, computes a random projection of the word-word similarity matrix, greatly reducing the time and space complexity [1]; however, as we have seen in Fig. 5 its cluster quality is not as good as that of HierNMF2/FlatNMF2.

Overall, FlatNMF2 is the best-performing method in our experiments, considering both cluster quality and efficiency. The relatively early software package CLUTO is also competitive [11].

### 7.5 Illustration

To visualize the tree of clusters/topics generated by HierNMF2, we show an illustration of the topic structure for a car review data set containing 954 articles in Fig. 8. First, we notice that the tree was not restrained to have a balanced structure, and HierNMF2 was able to determine

---

11 The run-time for CLUTO on Wiki-4.5M is absent: on our smaller system with 24 GB memory, it ran out of memory; and on our larger server with sufficient memory, the binary could not open a large data file (> 6 GB). The CLUTO software is not open-source and thus we only have access to the binary and are not able to build the program on our server.
the semantic organization on-the-fly. We can see that the reviews were clustered into commercial vehicles (‘cab’, ‘truck’, ‘towing’) versus regular vehicles, inexpensive cars versus luxury cars, sedans versus SUVs, in a hierarchical manner. Finally at the leaf level, HierNMF2 produced tight clusters such as sedans, hybrid cars, compact SUVs, minivans, luxury SUVs, and convertibles.

8 Conclusion

Clustering and topic modeling are among the major tasks needed in big data analysis due to the explosion of text data. Developing scalable methods for modeling large-scale text resources efficiently has become important for studying social and economic behaviors, significant public health issues, and network security to name a few.

In this paper we proposed HierNMF2 and FlatNMF2 for large-scale clustering and topic modeling. The proposed approaches are based on fast and cache-efficient algorithms for Rank-2 nonnegative matrix factorization that performs binary clustering and topic modeling, as well as an efficient decision rule for further splitting a leaf node in the hierarchy of topics. We further developed a custom routine for Sparse BLAS to accelerate sparse matrix multiplication, which is the main bottleneck in HierNMF2, FlatNMF2, and many other algorithms in data analytics. We evaluated the performance of HierNMF2 and FlatNMF2 on data sets with ground-truth labels and larger unlabeled data sets. HierNMF2 achieved similar topic quality compared to previous widely-used algorithms, but was more efficient by orders of magnitude. FlatNMF2 achieved better topic quality than all the other algorithms we compared with only a marginal computational overhead relative to HierNMF2. In summary, HierNMF2 and FlatNMF2 are over 100 times faster than NMF and about 20 times faster than LDA, and thus will have dramatic impacts on many fields requiring large-scale text analytics.

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,” in ICML ’13: Proc. of the 30th Int. Conf. on Machine Learning, 2013.

[2] S. Arora, R. Ge, R. Kannan, and A. Moitra, “Computing a nonnegative matrix factorization – provably,” in STOC ’12: Proc. of the 44th Symp. on Theory of Computing, 2012, pp. 145–162.

[3] V. Bittorf, B. Recht, C. Re, and J. Tropp, “Factoring nonnegative matrices with linear programs,” in Advances in Neural Information Processing Systems 25, ser. NIPS ’12, 2012, pp. 1214–1222.

[4] 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.

[5] D. M. Blei, “Probabilistic topic models,” Commun. ACM, vol. 55, pp. 77–84, 2012.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet allocation,” J. Mach. Learn. Res., vol. 3, pp. 993–1022, 2003.
[7] 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*, vol. 33, no. 8, pp. 1548–1560, 2011.

[8] A. Cichocki and A. H. Phan, “Fast local algorithms for large scale nonnegative matrix and tensor factorizations,” *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. E92A, no. 3, pp. 708–721, 2009.

[9] 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*, 2002, pp. 139–146.

[10] B. Drake, J. Kim, M. Mallick, and H. Park, “Supervised Raman spectra estimation based on nonnegative rank deficient least squares,” in *Proceedings 13th International Conference on Information Fusion, Edinburgh, UK*, 2010.

[11] A. Globerson, G. Chechik, F. Pereira, and N. Tishby, “Euclidean embedding of co-occurrence data,” *J. Mach. Learn. Res.*, vol. 8, pp. 2265–2295, 2007.

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

[13] K. Järvelin and J. Kekäläinen, “Cumulated gain-based evaluation of IR techniques,” *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, 2002.

[14] H. Kim and H. Park, “Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis,” *Bioinformatics*, vol. 23, no. 12, pp. 1495–1502, 2007.

[15] ——, “Nonnegative matrix factorization based on alternating non-negativity-constrained least squares and the active set method,” *SIAM J. on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 713–730, 2008.

[16] J. Kim, Y. He, and H. Park, “Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework,” *Journal of Global Optimization*, vol. 58, no. 2, pp. 285–319, 2014.

[17] J. Kim and H. Park, “Toward faster nonnegative matrix factorization: A new algorithm and comparisons,” in *ICDM ’08: Proc. of the 8th IEEE Int. Conf. on Data Mining*, 2008, pp. 353–362.

[18] ——, “Fast nonnegative matrix factorization: An active-set-like method and comparisons,” *SIAM J. on Scientific Computing*, vol. 33, no. 6, pp. 3261–3281, 2011.

[19] D. Kuang and H. Park, “Fast rank-2 nonnegative matrix factorization for hierarchical document clustering,” in *19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’13)*, 2013, pp. 739–747.

[20] D. Kuang, S. Yun, and H. Park, “SymNMF: Nonnegative low-rank approximation of a similarity matrix for graph clustering,” *J. Glob. Optim.*, 2014.
[21] A. Kumar, V. Sindhwani, and P. Kambadur, “Fast conical hull algorithms for near-separable non-negative matrix factorization,” in ICML ’13: Proc. of the 30th Int. Conf. on Machine Learning, 2013.

[22] C. L. Lawson and R. J. Hanson, Solving least squares problems. Englewood Cliffs, NJ: Prentice-Hall, 1974.

[23] D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” Nature, vol. 401, pp. 788–791, 1999.

[24] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, “Rcv1: A new benchmark collection for text categorization research,” J. Mach. Learn. Res., vol. 5, pp. 361–397, 2004.

[25] C.-J. Lin, “Projected gradient methods for nonnegative matrix factorization,” Neural Computation, vol. 19, no. 10, pp. 2756–2779, 2007.

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

[27] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, “Automating the construction of Internet portals with machine learning,” Inf. Retr., vol. 3, no. 2, pp. 127–163, 2000.

[28] P. Paatero and U. Tapper, “Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values,” Environmetrics, vol. 5, pp. 111–126, 1994.

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

[30] 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 information retrieval, 2003, pp. 267–273.