AFRL-RI-RS-TR-2018-173

FAST ALGORITHMS ON IMPERFECT, HETEROGENEOUS, DISTRIBUTED DATA

GEORGIA TECH RESEARCH CORPORATION

JULY 2018

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2018-173 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
NANCY A. ROBERTS
Work Unit Manager

/ S /
JON S. JONES
Technical Advisor, Information
Intelligence Systems and Analysis Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JULY 2018	FINAL TECHNICAL REPORT	AUG 2012 – FEB 2018

4. TITLE AND SUBTITLE

FAST ALGORITHMS ON IMPERFECT, HETEROGENEOUS, DISTRIBUTED DATA

5a. CONTRACT NUMBER
N/A

5b. GRANT NUMBER
FA8750-12-2-0309

5c. PROGRAM ELEMENT NUMBER
602702E

6. AUTHOR(S)

Haesun Park

5d. PROJECT NUMBER
XDAT

5e. TASK NUMBER
A0

5f. WORK UNIT NUMBER
07

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Georgia Tech Research Corporation
505 10th St NW
Atlanta, GA 30332-0001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RIEA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2018-173

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This project studies the design & implementation of several variants of NMF for text, graph & hybrid data analytics. It addresses challenges including solving new data analytics problems and improving the scalability of existing NMF algorithms. There are two major types of matrix representation of data: feature-data matrix and similarity matrix. Previous work showed successful application of standard NMF for feature-data matrix to areas such as text mining and image analysis, and Symmetric NMF (SymNMF) for similarity matrix to areas such as graph clustering & community detection. In this work, a divide-and-conquer strategy is applied to both methods to improve their time complexity from cubic growth with respect to the reduced low rank to linear growth, resulting in DC-NMF & HierSymNMF2 methods. Extensive experiments on large scale real world data show improved performance of these two methods. Furthermore, in this work NMF and SymNMF are combined into one formulation called Joint-NMF, to analyze hybrid data that contains both text content & connection structure information. They developed an open source software called SmallK (smallk.github.io) which offers several variants of NMF for fast clustering and topic modeling.

15. SUBJECT TERMS
Matrix factorization, large-scale data analytics, open source software, topic modeling, clustering, text mining and image analysis

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			NANCY A. ROBERTS
U	U	U	UU	50	19b. TELEPHONE NUMBER (*Include area code*)

Contents

List of Figures

List of Tables

1 Summary

Constrained low rank approximation (CLRA) is a general framework for data analysis, which usually has the advantage of being simple, fast, scalable and domain general. One of the most known constrained low rank approximation methods is nonnegative matrix factorization (NMF). This project studies the design and implementation of several variants of NMF for text, graph and hybrid data analytics. It addresses challenges including solving new data analytics problems and improving the scalability of existing NMF algorithms.

There are two major types of matrix representation of data: feature-data matrix and similarity matrix. Previous work showed successful application of standard NMF for feature-data matrix to areas such as text mining and image analysis, and Symmetric NMF (SymNMF) for similarity matrix to areas such as graph clustering and community detection. In this work, a divide-and-conquer strategy is applied to both methods to improve their time complexity from cubic growth with respect to the reduced low rank to linear growth, resulting in DC-NMF (Divide-and-Conquer NMF) and HierSymNMF2 (Hierarchical Symmetric NMF with Rank 2) methods. Extensive experiments on large scale real world data show improved performance of these two methods.

Furthermore, in this work NMF and SymNMF are combined into one formulation called JointNMF, to analyze hybrid data that contains both text content and connection structure information. Typical hybrid data where JointNMF can be applied includes paper/patent data where there are citation connections among content and email data and the sender/recipient relation is represented by a hypergraph and the email content is associated with hypergraph edges. An additional capability of the JointNMF is prediction of unknown network information which is illustrated using several real world problems such as citation recommendations of papers and activity/leader detection in organizations. We have developed an open source software called SmallK (smallk.github.io) which offers several variants of NMF for fast clustering and topic modeling.

2 Introduction

The amount of data and information has been quickly growing to a level that in many situations, data analytics and information retrieval can not be done without the help of computer algorithms. Advanced machine learning algorithms have been developed to process and understand the data. For example, to analyze texts, natural language models have been built to analyze grammars and syntax of sentences, and cognitive models were developed to make inferences based on the language structure. Those advanced models and methods are usually carefully designed with complicated assumptions/rules for a specific domain. They are powerful for many data analytics tasks. No matter what underlying model is behind it, much data can be summarized by some hidden patterns with much lower complexity. For example, all the sentences in an English encyclopedia share the same set of grammar rules, a long article can be categorized by a few topic words, a complicated image reduced to 256 colors is still identifiable by a human, etc. Constrained low rank approximation is a category of methods that try to find out the low-complexity patterns behind matrix/tensor encodable data, without complicated assumptions about the model behind the data. In many situations, the hidden patterns discovered by low rank approximation methods give us enough valuable information about the data. Due to the simplicity and independence of the underlying model, low rank approximation methods are widely applicable and easier to scale. For

example, nonnegative matrix factorization (NMF), a low rank approximation method with nonnegative constraints, has been applied to document clustering [1], image analysis [2], cancer subtype detection [3], blind source separation for audio [4], and many other areas. Some advantages of NMF based algorithms are: (1) good interpretability (For example, NMF based methods not only give clustering assignments, the generated nonnegative basis vectors also summarize each cluster very well.), (2) being supported by fast numerical routines and sophisticated numerical libraries, (3) ability to utilize scalable MPI based implementations [5, 6]. This project focuses on NMF and its variants for clustering of text, graph and hybrid data where text and graph are merged.

We seek to harness fundamental advances in machine learning and data analytics to enable inquiry and discovery from large volume, high dimensional, distributed, heterogeneous, streaming, and time-varying data. This project focuses on the algorithms and infrastructures for efficient operation in a general setting. Our approach involves developing fast and scalable algorithms, informative data representation, transformation, and analysis techniques. A fully open-source software framework called SmallK for a variety of data analytics tasks is developed.

The key issues we investigate are: 1. The design and implementation of fast algorithms for constrained low rank approximations on a variety of computing environment from laptop to parallel and distributed systems for the development of efficient open-source software implementations. 2. Evaluation of the convergence behavior and quality of solutions of proposed algorithms and analysis of runtime behavior. 3. Evaluation and applications of our algorithms for large-scale real-life data sets and demonstration of effectiveness relative to existing approaches.

We provide a general framework based on constrained matrix lower-rank approximation, which is flexible to various problem needs while keeping the main structure for scalability for very large-volume high-dimensional data analysis problems such as clustering, classification, dimension reduction, topic modeling, and graph understanding. The open-source software package SmallK in C++ and Python makes the developed algorithms and theory widely accessible to a broad community of researchers and practitioners.

Many important problems in large-scale data analytics can be modeled as matrix lower-rank approximation problems. They include approaches such as dimension reduction, clustering, topic modeling, and graph clustering. We show that by applying different constraints to the general framework of matrix lower-rank approximation, and especially by using some variations of the nonnegative matrix factorization (NMF), we can model these problems and design efficient and scalable algorithms to solve real-life problems of large volume, high dimension, and with missing components. Compared to the existing approaches such as those based on statistical methods, our problem modeling heavily relies on matrix computation and numerical optimization. The impacts include maximum efficiency and scalability for large-scale high-dimensional problems, higher quality in the computed solutions, as well as flexibility to incorporate various constraints that reflect the problem situation realistically, using one framework with the capability to adapt to different solution needs.

3 Methods, Asumptions, and Procedures

We assume the data has nonnegative matrix/vector representation $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}_+^{m \times n}$, where $\mathbb{R}_+$ is the set of all nonnegative real numbers. Nonnegative matrix factorization approximates X as a conic combination of k nonnegative basis vectors, where usually $k \ll \min\{m,n\}$. We assume

the k basis vectors are stored in a matrix $W = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n] \in \mathbb{R}^{m \times k}$ and the nonnegative combination coefficients are stored in a matrix $H = (h_{ij}) \in \mathbb{R}^{k \times n}$ such that each $\boldsymbol{x}_j$ is approximated by $\sum_{i=1}^{k} h_{ij}\boldsymbol{w}_i$, or equivalently, the matrix X is approximated by the product WH. Intuitively, if X is well approximated, the column vectors of W will be good representative vectors of the entire data set, and the coefficients in H will show the proportion of each components in each data item. When we have enough representative vectors, most data items will be mainly associated with one representative vector, and therefore a clustering assignment can be induced. More specifically, data $\boldsymbol{x}_j$ belongs to cluster i if $h_{ij} > h_{lj}$ for all $l \neq i$ and $\boldsymbol{w}_i$ is the cluster representative of cluster i. To ensure the quality of such clustering and representative vectors, we would want the approximation error to be as small as possible. The two most common error measures for NMF are Frobenius distance and Kullback-Leibler divergence [7]. In this project we only use Euclidean distance which defines NMF as the optimization problem in Equation (1)

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

because it has the advantages including flexibility for designing efficient and scalable algorithms for large-scale problems, ability to produce more accurate solutions in a variety of noisy real-life applications even when other measures such as KL-divergence can model the problems better theoretically [8, 9, 10, 11, 12, 13], convenience to combine low rank approximations of multiple matrices and to add certain regularization terms.

3.1 Clustering of Text and Graph

Many types of data can be encoded as nonnegative matrix, such as text, graph, image, hyperspectral image, sound, etc. Although NMF does have applications on all those types of data, this project focuses on text and graph data.

There are two major types of matrix representation of data: feature-data matrix and similarity matrix. Typically, text data are usually encoded as feature-data matrix, where the features are usually a subset of words that appears in the data set, possibly with some transformations such as stop-word removal, stemming, lemmatization, etc. The most straightforward encoding is perhaps term-frequency matrix, where each entry X_{ij} are integers representing the number of appearances of the i-th word in the j-th document. For better clustering quality, one usually needs to apply some normalizations such as TF-IDF [14] and column normalization. Clustering using a feature-data matrix is called feature based clustering, for which the most famous example is K-means [15]. The standard NMF (1) is also a good clustering algorithm for nonnegative feature-data matrices. Particularly, it is shown to be an effective method for text clustering and topic modeling [10].

Sometimes, we only know the relation between data items in a space without knowing their actual vector representation. For example, for kernel methods we only know the inner product or data items in the kernel space; for graph data, the only given information is the connection relations between data items. These relation info can usually be encoded in a symmetric similarity matrix $S \in \mathbb{R}^{n \times n}$, where S_{ij} measures the "strength" of the relation between the i-th and j-th data item. One of the most important types of data that can be represented by a nonnegative symmetric matrix is graph. Besides abundant real-world graphs such as social networks, citation networks, web-linkage networks and communication networks, there are also many graphs designed to model certain

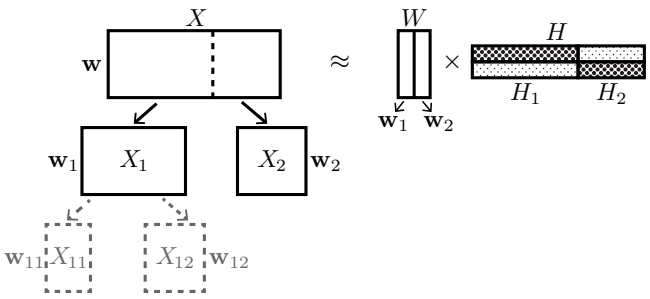


Figure 1: Illustration of how DC-NMF use divide-and-conquer to go from rank-2 NMF to higher rank NMF. The dark part in H means relative larger values.

relations such as product co-purchasing network, co-author network, etc. In some situations, such as when data points are embedded in a nonlinear manifold, it is better to transform featured-based data into a graph (for example, using k-nearest neighbors) and perform graph clustering. One of the famous graph clustering algorithms is the spectral clustering algorithm [16].

3.2 Fast NMF based on Divide-and-Counquer

We propose a fast algorithm for computing NMF for any given $k \geq 2$, which we call DC-NMF (Divide-and-Conquer NMF). Based on the fast rank-2 NMF algorithm and a divide-and-conquer method, DC-NMF computes a high quality W for NMF. We also provide and compare several alternative formulations for DC-NMF.

The value of k represents the number of clusters or number of topics, which is often larger than 2. In addition, since a larger k value produces a better low rank approximation, a fast algorithm that works for $k > 2$ is needed. Increasing the reduced rank k in the unconstrained low rank approximation (SVD) strictly improves the approximation quality until k reaches $\text{rank}(X)$ [17].

Unlike for the SVD, one cannot use successive rank-1 deflations to go from rank-2 NMF to rank-k NMF for $k > 2$ [18]. For NMF, all vectors in $W \in \boldsymbol{R}^{m\times k}$ typically change completely when the reduced rank k changes. However, since rank-2 NMF can be used for binary clustering, the columns of X can be divided into two clusters based on H from rank-2 NMF, forming two submatrices X_1 and X_2, as illustrated in Figure 1. Assume we have a rank-2 NMF of X as $X \approx \text{span}_+(\boldsymbol{w}_1, \boldsymbol{w}_2)$. Then we view $\boldsymbol{w}_i$ as a representative vector for X_i, i.e., $X_i \approx \text{span}_+(\boldsymbol{w}_i)$, for $i = 1, 2$. If $\text{rank}_+(X) > 2$, then we can obtain a better approximation of X by replacing one of $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ with two basis vectors obtained by applying rank-2 NMF on X_1 or X_2. Our cluster tree traversing rule determines this next submatrix for which we increase the reduced rank for NMF approximation from 1 to 2, so that the overall nonnegative approximation error for X is locally reduced the most.

The matrix X (after a proper permutation of columns), the partition $X_1, \ldots, X_k$ and the representative vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k$ can be used as the columns of W and we can obtain H from one step of NLS $\min_{H\geq 0} \|WH - X\|_F$ to get W and H as the NMF solution. We can also perform several more NLS iterations for NMF to further reduce the approximation error. The stopping criteria for flat NMF can be used here, for example the one used in [19] that checks whether the solution is a stationary point. In practice, we have found that there is usually a significant drop of approximation error after one full alternating iteration of computing H and then updating W, and subsequent iterations did not significantly reduce the approximation error. Therefore,

Table 1: Various priority scores for choosing a cluster to split.

Name	Formula	Need Pre-split	Note
Score 0	$s = \min_{\tilde{\boldsymbol{h}}} \|\tilde{X} - \tilde{\boldsymbol{w}}\tilde{\boldsymbol{h}}^\mathsf{T}\|_F^2 - \sum_{i=1}^{2} \min_{\tilde{\boldsymbol{h}}_i} \|\tilde{X}_i - \tilde{\boldsymbol{w}}_i\tilde{\boldsymbol{h}}_i^\mathsf{T}\|_F^2$	Y	The score proposed in this paper
Score 1	$s = \min_{\boldsymbol{u},\boldsymbol{v}} \|\tilde{X} - \boldsymbol{u}\boldsymbol{v}^\mathsf{T}\|_F^2 - \sum_{i=1}^{2} \min_{\boldsymbol{u}_i,\boldsymbol{v}_i} \|\tilde{X}_i - \boldsymbol{u}_i\boldsymbol{v}_i^\mathsf{T}\|_F^2$	Y	The score used for hierarchical clustering in [12]
Score 2	$s = \text{mNDCG}(\tilde{\boldsymbol{w}}_1) \times \text{mNDCG}(\tilde{\boldsymbol{w}}_2)$	Y	The score used for hierarchical topic modeling in [10]
Score 3	$s_i = \min_{\tilde{\boldsymbol{h}}} \|\tilde{X}_i - \tilde{\boldsymbol{w}}_i\tilde{\boldsymbol{h}}^\mathsf{T}\|_F^2$	N	Measures how well $\tilde{X}_i$ is represented by $\tilde{\boldsymbol{w}}_i$.
Score 4	$s_i = \min_{\boldsymbol{u},\boldsymbol{v}} \|\tilde{X}_i - \boldsymbol{u}\boldsymbol{v}^\mathsf{T}\|_F^2$	N	Measures how close $\tilde{X}_i$ is to a rank-1 matrix.
Score 5	$s_i = \|\tilde{X}_i - \tilde{W}\tilde{H}_i\|_F^2$	N	Measures how well $\tilde{X}_i$ is represented by $\tilde{W}$.

in our proposed DC-NMF, we perform one iteration to compute H and update W starting with W given by HierNMF2 [10], in order to obtain a good solution while maintaining the speed advantage. The approximation error $\|X - WH\|_F^2$ can be computed by the formula $\|X - WH\|_F^2 = \|X\|_F^2 - 2 \cdot \text{trace}(HX^\mathsf{T}W) + \text{trace}(W^\mathsf{T}WHH^\mathsf{T})$ to avoid directly computing $X - WH$, which is computationally expensive and can destroy the sparse structure of X.

The priority scores for DC-NMF proposed in [10, 12] need to pre-split a cluster (of columns) in order to compute a priority score. We can also define heuristic scores that do not need a pre-split. For example, supposing $\tilde{X}$ is a submatrix corresponding to a cluster and $\tilde{\boldsymbol{w}}$ is its representative vector, we can define a heuristic score as $\min_{\tilde{\boldsymbol{h}}} \|\tilde{X} - \tilde{\boldsymbol{w}}\tilde{\boldsymbol{h}}^\mathsf{T}\|_F$ to check how well $\tilde{X}$ is represented as rank-1 matrix $\tilde{\boldsymbol{w}}\tilde{\boldsymbol{h}}^\mathsf{T}$ i.e., how coherent its columns are, and split (i.e. approximate by rank 2) the worst represented cluster. We summarize some of the priority scores in Table 1, where $\tilde{\boldsymbol{h}}$, $\boldsymbol{u}$ and $\boldsymbol{v}$ are column vectors of proper size. In our experiments, we found that Score 0 and Score 1 often obtain significantly lower approximation errors than the other scores. However, Score 1 requires significantly longer computation time than Score 0 since Score 1 also computes a rank-1 SVD.

3.3 SymNMF for Similarity/Connection-Based Clustering

The similarity/connection information can usually be encoded in a nonnegative symmetric matrix. To utilize the symmetric structure, we need a variant of NMF called Symmetric NMF (SymNMF) [11], the formulation of which is

$$\min_{H \geq 0} \|S - H^\mathsf{T}H\|_F \tag{2}$$

where $S \in \mathbb{R}_+^{n \times n}$ is a nonnegative symmetric matrix, $H \in \mathbb{R}_+^{k \times n}$ and $k \ll n$. Here we assume S is the matrix representation of a graph since most similarity/connection relations can be modeled by a graph. We assume the graph under study is $\mathcal{G} = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ and $E \subset V \times V$. The matrix $S = (w_{ij})$ is the adjacency matrix of graph $\mathcal{G}$, where $w_{ij} = w(v_i, v_j)$ is the weight of edge (v_i, v_j). Some choices of the input matrix S for SymNMF are the adjacency matrix $S^\mathcal{G}$ and

Algorithm 1 Divide-and-Conquer Framework for Divisive Hierarchical Clustering

1: **Initialization**: One cluster containing all nodes.
2: **repeat**
3: Choose one of the clusters to split.
4: Split the chosen cluster into two clusters.
5: **until** there are k clusters (or other stopping criteria)

the normalized adjacency matrix $D^{-1/2}S^{\mathcal{G}}D^{-1/2}$, where $D = \text{diag}(d_1, \ldots, d_n)$ and $d_i = \sum_{j=1}^{n} S_{ij}^{\mathcal{G}}$ is the degree of node i. When S is the adjacency matrix, (2) is a relaxation of maximizing the ratio association; when S is the normalized adjacency matrix, (2) is a relaxation of minimizing the normalized cut [20].

3.4 Hierarchical SymNMF for Large Scale Community Detection

The algorithm we introduce uses a similar divide-and-conquer idea, as summarized in where a cluster is a community, and the task of splitting a community is performed by our rank-2 version of SymNMF. The decision to choose the next node to split is based on a criteria discussed in the next section. In the following sections, we denote S as the similarity matrix representing a graph $\mathcal{G}$, and S_c as the matrix representation of a community, i.e., a subgraph of $\mathcal{G}$ (the corresponding submatrix of S).

3.4.1 Splitting a Community Using Rank-2 SymNMF

Splitting a community is achieved by rank-2 SymNMF of $S_c \approx H^{\intercal}H$ where $H \in \mathbb{R}_{+}^{2\times n}$. The result H naturally induces a binary split of the community: suppose $H = (h_{ij})$, then

$$c_i = \begin{cases} 1, & h_{1i} > h_{2i}; \\ 0, & \text{otherwise.} \end{cases}$$

where c_i is the community assignment of the ith graph node.

A formal formulation of rank-2 SymNMF is the following optimization problem:

$$\min_{H \geq 0} \|S - H^{\intercal}H\|_F^2 \tag{3}$$

where $H \in \mathbb{R}_{+}^{n\times 2}$. This is a special case of SymNMF when $k = 2$, which can be solved by a general SymNMF algorithm [20, 11]. However, by combining the *alternating nonnegative least squares* (ANLS) algorithm for SymNMF from [11] and the fast algorithm for rank-2 NMF from [10], we can obtain a fast algorithm for rank-2 SymNMF.

First, we rewrite (3) into asymmetric form plus a penalty term [21]:

$$\min_{W,H \geq 0} \|S - W^{\intercal}H\|_F^2 + \alpha\|W - H\|_F^2 \tag{4}$$

where W and $H \in \mathbb{R}_{+}^{n\times 2}$ and $\alpha > 0$ is a scalar parameter for the tradeoff between the approximation

error and the difference between W and H. Formulation (4) can be solved using a two-block coordinate descent framework, alternating between the optimization for W and H. When we solve for W, (4) can be reformulated as

$$\min_{W^\intercal \geq 0} \left\| \begin{bmatrix} H^\intercal \\ \sqrt{\alpha} I_2 \end{bmatrix} W - \begin{bmatrix} S \\ \sqrt{\alpha} H \end{bmatrix} \right\|_F^2 \tag{5}$$

where I_2 is the 2×2 identity matrix. Similarly, when we solve for H, (4) can be reformulated as

$$\min_{H \geq 0} \left\| \begin{bmatrix} W^\intercal \\ \sqrt{\alpha} I_2 \end{bmatrix} H - \begin{bmatrix} S \\ \sqrt{\alpha} W \end{bmatrix} \right\|_F^2 \tag{6}$$

3.4.2 Choosing a Node to Split Based on Normalized Cut

The "best" community to split further is chosen by computing and comparing *splitting scores* for all current communities corresponding to the leaf nodes in the hierarchy. The proposed splitting scores are based on normalized cut. We make this choice because: 1) normalized cut determines whether a split is structurally effective since it measures the difference between intra- and inter-connections among network nodes; 2) for SymNMF, when S is the normalized adjacency matrix, the SymNMF objective function is equivalent to (a relaxation of) minimizing the normalized cut which is the preferred choice in graph clustering [11].

We illustrate our splitting criteria using an example graph shown in Figure 2. It originally has three communities P_1, P_2 and P_3, and the corresponding normalized cut is

$$\begin{aligned} \text{ncut}(P_1, P_2, P_3) = &\frac{\text{out}(P_1)}{\text{within}(P_1) + \text{out}(P_1)} + \frac{\text{out}(P_2)}{\text{within}(P_2) + \text{out}(P_2)} \\ &+ \frac{\text{out}(P_3)}{\text{within}(P_3) + \text{out}(P_3)} \end{aligned}$$

The community P_3 is now split into two smaller communities Q_1 and Q_2 and normalized cut can be used to measure the goodness of this split. We consider three possibilities: (1) Isolate P_3 and compute normalized cut of the split as

$$\text{ncut}\big|_{P_3}(Q_1, Q_2) = \frac{\text{out}\big|_{P_3}(Q_1)}{\text{within}(Q_1) + \text{out}\big|_{P_3}(Q_1)} + \frac{\text{out}\big|_{P_3}(Q_2)}{\text{within}(Q_2) + \text{out}\big|_{P_3}(Q_2)}$$

where the subscript P_3 means only consider the edges inside P_3. We denote the above criterion by `ncut_local`. (2) A more global criterion is to also consider the edges that go across P_3:

$$\text{ncut}(Q_1, Q_2) = \frac{\text{out}(Q_1)}{\text{within}(Q_1) + \text{out}(Q_1)} + \frac{\text{out}(Q_2)}{\text{within}(Q_2) + \text{out}(Q_2)}$$

This criterion is denoted by `ncut_global`. (3) Minimize the global normalized cut using a greedy strategy. Specifically, choose the split that results in the minimal increase in the global normalized

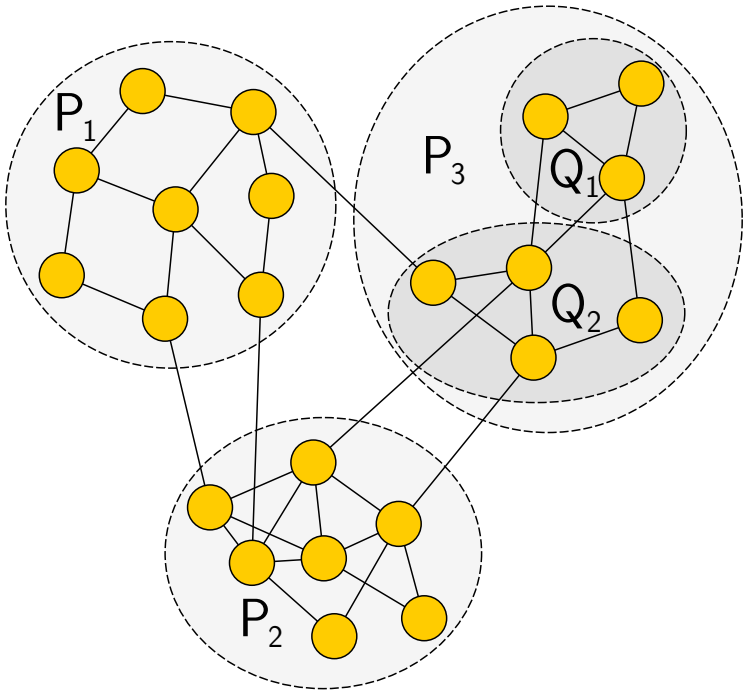


Figure 2: A graph for illustrating the splitting criteria for HierSymNMF2. The structure of the graph is inspired by Figure 1 from [22].

cut:

$$\begin{aligned}&\text{ncut}(P_1,P_2,Q_1,Q_2) - \text{ncut}(P_1,P_2,P_3)\\ &= \frac{\text{out}(Q_1)}{\text{within}(Q_1)+\text{out}(Q_1)} + \frac{\text{out}(Q_2)}{\text{within}(Q_2)+\text{out}(Q_2)} - \frac{\text{out}(P_3)}{\text{within}(P_3)+\text{out}(P_3)}\end{aligned}$$

We denote this criterion by `ncut_global_diff` and will compare the performance of these three criteria in later sections.

3.5 Hybrid Clustering Using JointNMF

DC-NMF and HierSymNMF2 have good performance on text clustering and graph clustering, respectively. There are also numerous data sets containing both text content and connection structure. For example, in a data set of research papers or patents, papers or patents have text content where the citations or co-author relationships define the connection structure; in a data set of emails, email messages have text content and the sender-recipient relations define a hypergraph structure where one email may have multiple recipients. When the connection structure is represented as edges in a graph, in the former case the text content is associated with graph nodes while in the latter case the text content is associated with hypergraph edges. A hybrid clustering method is designed to utilize both content and connection structure information, thus taking advantage of the full information provided in the data. Since NMF for content clustering and SymNMF for graph clustering have

the same underlying matrix factorization framework, they can be merged at the objective function level, becoming the JointNMF.

First we assume that the text content is associated with the graph nodes. For example, a collection of research papers or patents can be represented in a graph where the content information of each paper or patent is a graph node and the citation information provides the graph connection information. We assume that a data set's text information is represented in a nonnegative matrix $X \in \mathbb{R}_+^{m\times n}$ and the graph structure is represented in a nonnegative symmetric matrix $S \in \mathbb{R}_+^{n\times n}$, where m is the number of features, and n is the number of data items.

The hybrid clustering method we propose finds a low rank representation that simultaneously represents the text content and the graph structure of the data items by jointly optimizing the combined NMF (1) and SymNMF (2) objective functions:

$$\min_{W\geq 0, H\geq 0} \alpha_1 ||X - WH||_F^2 + \alpha_2 ||S - H^\intercal H||_F^2. \tag{7}$$

where $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$ are the weighting parameters. By adjusting the parameters α_i, we can emphasize one over the other. In the extreme case, some α_i can be set to zero: e.g. when $\alpha_2 = 0$ in the above, we are only concerned with the content, when $\alpha_1 = 0$, we only pay attention to the structural information and ignore the content. Excluding these special cases, we can assume $\alpha_1 = 1$ without loss of generality and Eqn. (7) becomes

$$\min_{W\geq 0, H\geq 0} ||X - WH||_F^2 + \alpha ||S - H^\intercal H||_F^2. \tag{8}$$

with $\alpha \geq 0$ as the weighting parameter.

Now we extend our method to hypergraphs where the text content is associated with hypergraph nodes. Once this is done, it would be natural to extend our method further to the cases where text is associated with graph or hypergraph edges due to the duality that exists between edges and nodes of a hypergraph and the fact that a graph can be treated as a special case of a hypergraph.

There are many ways to find a solution for the objective function (8). Theoretically, a Newton-like algorithm can be developed to directly solve (8). However, as pointed out in [11], a Newton-like algorithm can not utilize the sparsity of X and S for speeding up because the matrices $X - WH$ and $S - H^\intercal H$ need to be computed explicitly and thus the sparsity will be destroyed. On the other hand, an alternating nonnegative least square (ANLS) algorithm can be sped up with sparsity. To apply an ANLS-like algorithm that can utilize the sparse nature of text documents and associated networks, we propose reformulating (8) in the following form with a penalty term

$$\min_{W,H,\tilde{H}\geq 0} ||X - WH||_F^2 + \alpha ||S - \tilde{H}^\intercal H||_F^2 + \beta ||\tilde{H} - H||_F^2. \tag{9}$$

where $\tilde{H} \in \mathbb{R}_+^{k\times n}$ and $\beta \geq 0$ is the regularization parameter. This reformulation is motivated from our earlier work to generate an algorithm that is based on the block coordinate descent (BCD) scheme so that each sub-problem in the BCD is a nonnegativity constrained least squares (NLS) problem for which we have developed a highly efficient algorithm and optimized open-source software [5]. Then Eqn. (9) can be solved using a 3-block coordinate descent (BCD) scheme, i.e. minimize the objective function with respect to W, $\tilde{H}$ and H in turn. Specifically, we solve the

following three subproblems in turn:

$$\min_{W\geq 0} \|H^\mathsf{T}W^\mathsf{T} - X^\mathsf{T}\|_F \tag{10}$$

$$\min_{\tilde{H}\geq 0} \left\| \begin{bmatrix} \sqrt{\alpha}H^\mathsf{T} \\ \sqrt{\beta}I_k \end{bmatrix} \tilde{H} - \begin{bmatrix} \sqrt{\alpha}S \\ \sqrt{\beta}H \end{bmatrix} \right\|_F \tag{11}$$

$$\min_{H\geq 0} \left\| \begin{bmatrix} W \\ \sqrt{\alpha}\tilde{H}^\mathsf{T} \\ \sqrt{\beta}I_k \end{bmatrix} H - \begin{bmatrix} X \\ \sqrt{\alpha}S \\ \sqrt{\beta}\tilde{H} \end{bmatrix} \right\|_F \tag{12}$$

where each subproblem is simply a nonnegative least squares problem (NNLS). The above three block BCD algorithm converges to a stationary point according to Bertsekas' theorem [23]. The identity submatrices I_k in the above equations make the problem better conditioned than the subproblems in the standard NMF that uses two block BCD alternating updating W and H. We solve each NLS problem using the block principal pivoting (BPP) algorithm [19]. Theoretically, to force H to be identical to $\tilde{H}$, the value of the parameter β has to be infinity. This problem has been studied extensively and we use a scheme similar to that proposed in [24]. It should be pointed out that in [19] it is shown that algorithms based on the BCD framework have guaranteed convergence to a stationary point, whereas, popular and easy to implement algorithms such as Multiplicative Updating (MU) may not converge. In addition, extensive experiments show that the BPP method is faster and more accurate than MU.

4 Results and Discussions

4.1 DC-NMF Experiments

In this section, we show experimental results for DC-NMF and compare it with state-of-the-art algorithms for NMF, clustering, and topic modeling. First, we focus on the role of DC-NMF as a generic algorithm for computing NMF and evaluate its runtime versus approximation error. Then, we apply DC-NMF to small- to medium-scale data sets with ground-truth to evaluate its effectiveness for clustering before moving to much larger data sets for the benchmarking of computational efficiency. Our experiments were run on a server with two Intel E5-2620 processors, each having six cores, and 377 GB memory.

Before proceeding to the experimental results, we first describe the data sets and experimental settings in detail.

4.1.1 Data Sets

Six text data sets were 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 have a

[1] `http://www.daviddlewis.com/resources/testcollections/reuters21578/` (retrieved in June 2014)

[2] `http://qwone.com/~jason/20Newsgroups/` (retrieved in June 2014)

Table 2: 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)

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[2], we observed that many articles had duplicate paragraphs due to cross-referencing. We discarded cited paragraphs and signatures, which increased the difficulty of clustering. 3. **Cora** [25] 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 447 papers from the 2001-2003 period [26], which were associated with labels indicating the technical area (algorithms, learning theory, vision science, etc). 5. **RCV1** [27] 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**[3] 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 2. The first four medium-scale data sets have ground-truth labels for the evaluation of cluster quality, while the remaining two large scale data sets are treated as unlabeled. 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 [1].

4.1.2 Implementation

We implemented DC-NMF both in Matlab and in an open-source C++ software library called `SmallK`[4] [5]. The existing methods we compared DC-NMF with are grouped into three categories: NMF algorithms, clustering methods and topic modeling methods. Though clustering and topic modeling can be unified in the framework of matrix factorization, we label a method as belonging to one of the two categories according to the task for which it was originally targeted.

NMF Algorithms. We compared the following algorithms for computing rank-k NMF:[5]

[3] `https://dumps.Wikimedia.org/enWiki/`

[4] `https://smallk.github.io/`

[5] Besides the listed algorithms, we also experimented with a recent algorithm based on coordinate descent with a greedy rule to select the variable to improve at each step [28]. However, this algorithm became increasingly slow when

- `MU`: The multiplicative update algorithm for Frobenius-norm based NMF [7]. `MU` is not guaranteed to converge to a stationary point solution although it reduces the objective function after each iteration.

- `ANLS/BPP`: The block principal pivoting algorithm that follows the two-block coordinate descent framework [29, 19]. We will often refer to this method as simply BPP.

- `HALS/RRI`: The hierarchical alternating least squares algorithm [30, 21], which is a $2k$-block coordinate descent method. We will simply refer to this as `HALS`.

Many schemes that can be used to accelerate the above algorithms have been proposed in the literature (e.g. [31, 32]) but our comparisons are on the above baseline algorithms.

Clustering Methods. The clustering methods we compared include:

- `nmf-hier`: Hierarchical clustering based on standard NMF with ANLS and an active-set method for NLS [33]. 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. The block principal pivoting (BPP) method [29, 19] is used as an exemplar algorithm to solve the NLS subproblems. In our experiments, multiplicative update rule algorithms [2] 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 [10].

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

- `CLUTO`: A clustering toolkit[6] written in C++. We used the default method in its `vcluster` program, namely a repeated bisection algorithm.

Topic Modeling Methods. The topic modeling methods we compared include:

- `Mallet-LDA`: The software MALLET[7] 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 [34]. 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 [34]. The software is written in Java[8]. We used the default parameters.

- `XRAY`: Another recent algorithm to solve NMF with separability constraints [35]. 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.

we increased k and kept the size of A the same. Therefore, we did not include it in our final comparison.

[6] `http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview`

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

[8] `https://github.com/mimno/anchor`

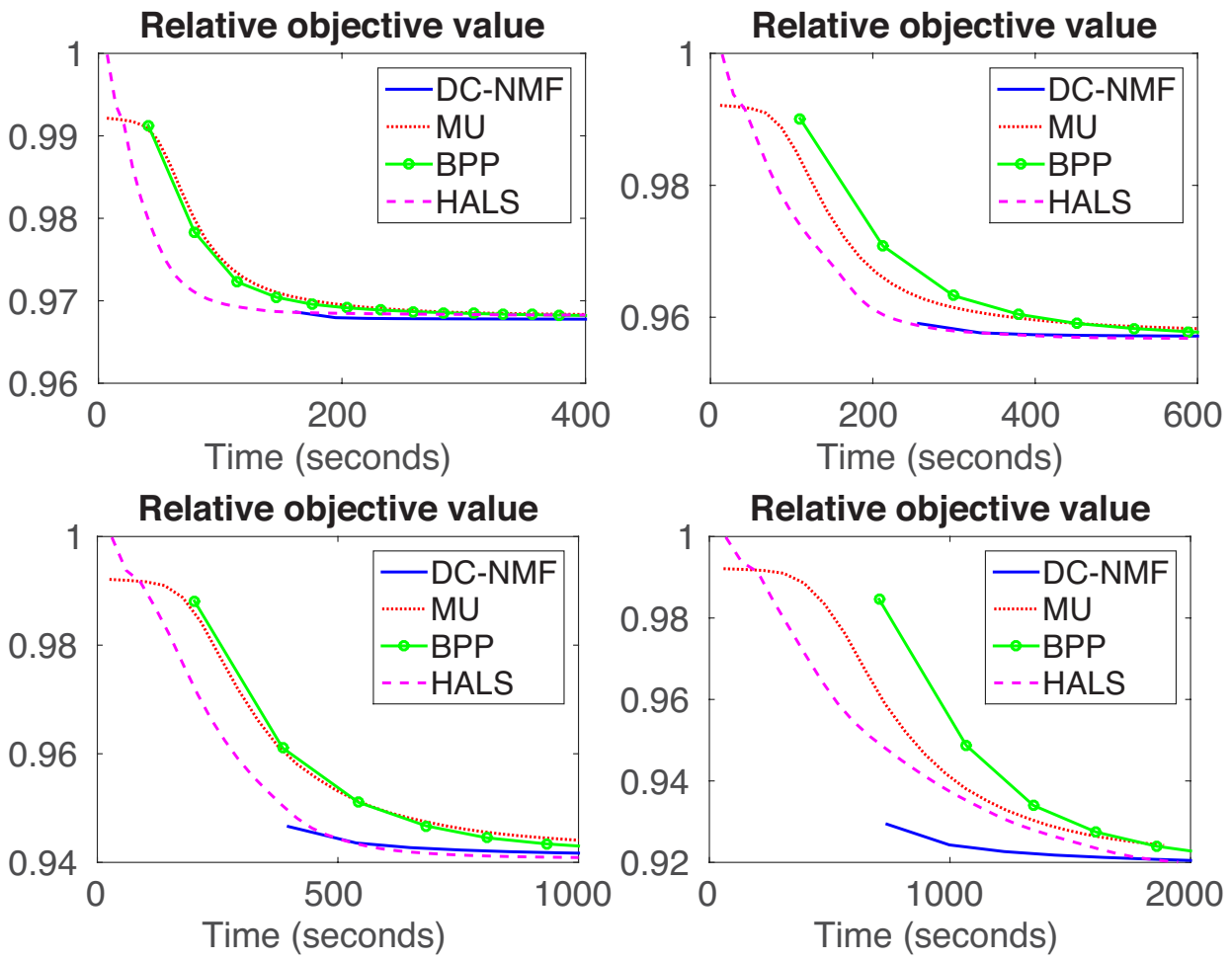


Figure 3: Comparison of approximation error between DC-NMF versus other algorithms for computing NMF. Results are shown for $k = 20, 40, 80, 160$.

- `Hottopixx`: A recent method that formulates Separable NMF as a linear program and solves it using incremental gradient descent [36]. We used the default parameters.

4.1.3 Experimental Settings

To evaluate the cluster and topic quality, we use the *normalized mutual information* (NMI). 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, one for each level. When evaluating the results given by DC-NMF, 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, 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 with random initializations. Average measurements are

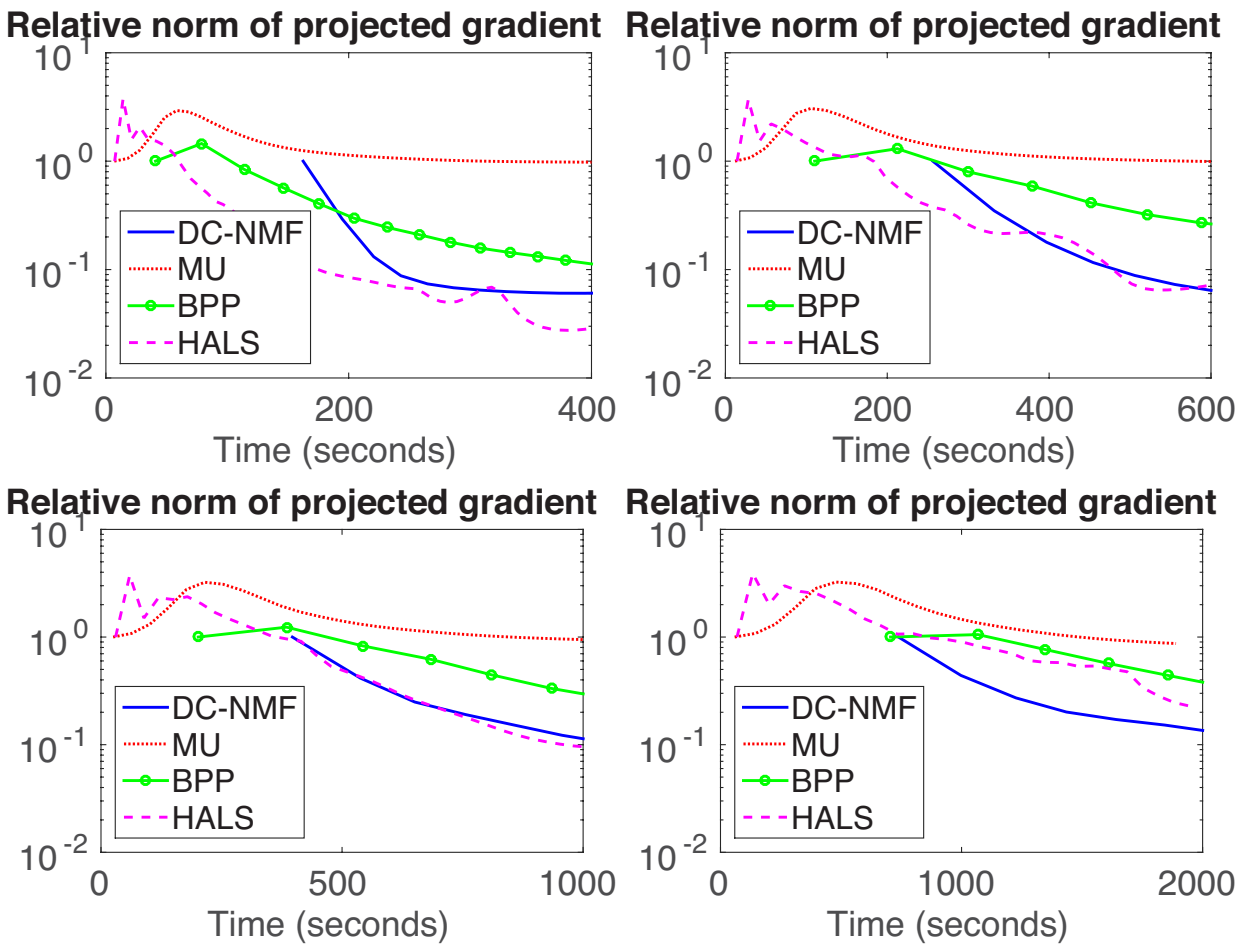


Figure 4: Comparison of projected gradient norm between DC-NMF versus other algorithms for computing NMF. Results are shown for $k = 20, 40, 80, 160$.

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 labels at the deepest level for labeled data sets (see Table 2); and we set $c = 60$ for **RCV1** and $c = 80$ for **Wiki-4.5M**. 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[9]. We use both phases for data sets with fewer than 20,000 documents, and only the batch-update phase for data sets with more than 20,000 documents. For NMF, we use the projected gradient norm as the stopping criterion [37] with a tolerance parameter $\varepsilon = 10^{-4}$. The projected gradient norm is sensitive to the scaling of the W and H factors: WD and $D^{-1}H$ yield the same approximation error but different values of projected gradient norm, where D is a diagonal matrix with positive entries on the diagonal (see details in [19, 38]). To ensure a fair comparison between different methods, before computing a projected gradient norm, we make the columns of W have unit 2-norm and scale H accordingly. All the methods are implemented with multi-threading.

[9] `http://math.ucla.edu/~dakuang/software/kmeans3.html`

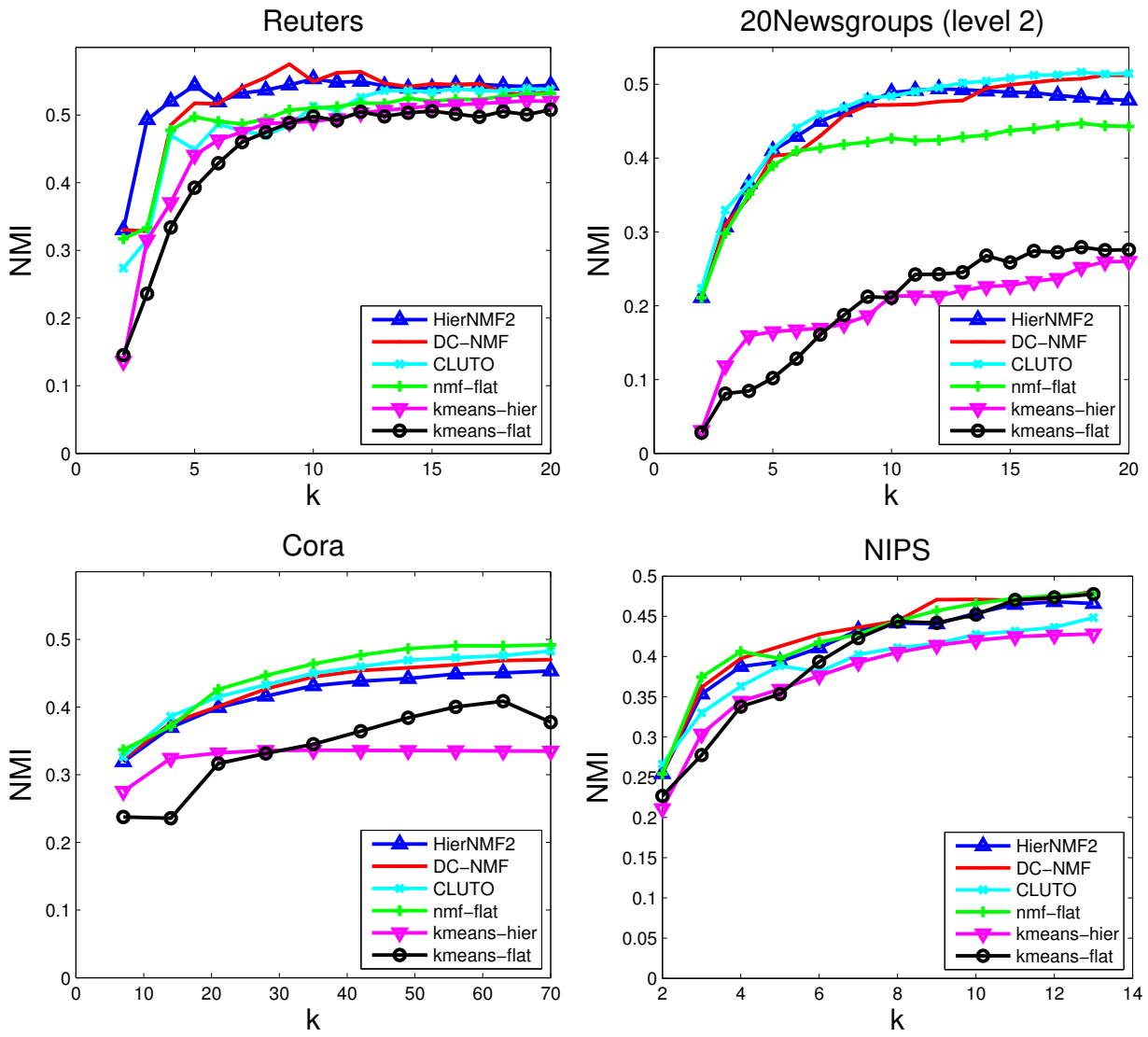


Figure 5: DC-NMF versus other *clustering methods* in cluster quality evaluated by normalized mutual information (NMI).

Figs. 3 and 4 show the performance of DC-NMF, MU, BPP, and HALS algorithms in terms of relative objective function value vs. time in seconds, and relative norm of the projected gradient vs. time in seconds, respectively.

4.2 DC-NMF for Clustering and Topic Modeling

4.2.1 Cluster Quality

Figs. 5 and 6 show the cluster quality on four labeled data sets, comparing DC-NMF with the state-of-the-art *clustering methods* and *topic modeling methods*, respectively. `nmf-hier` generates the identical results with DC-NMF (but the former is less efficient) and is not shown in Fig. 5.

We can see that DC-NMF gives better cluster and topic quality in many cases, and improves

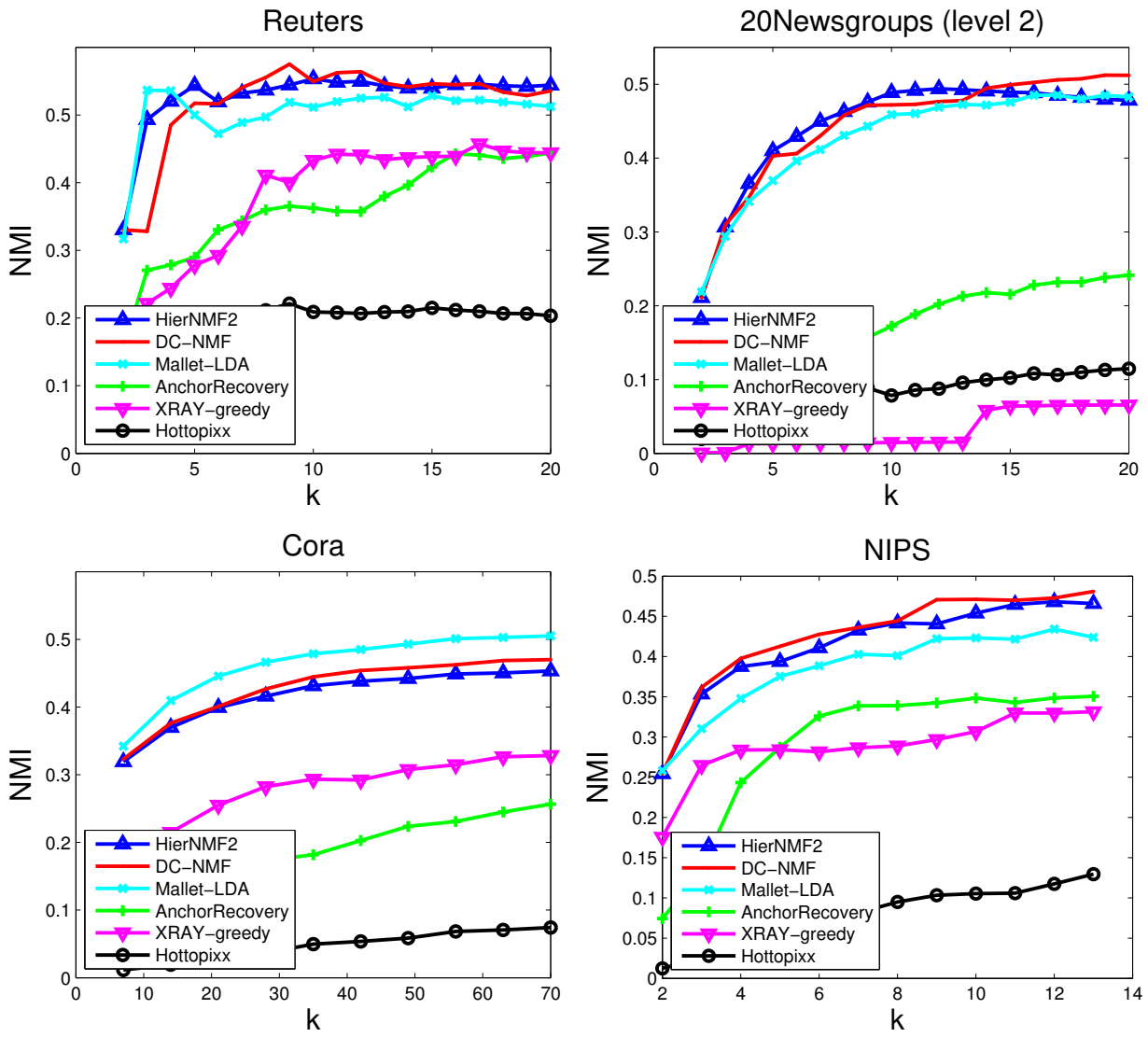


Figure 6: DC-NMF versus other *topic modeling methods* in cluster quality evaluated by normalized mutual information (NMI).

the performance of HierNMF2 in every case. One possible reason for the better performance of DC-NMF 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.

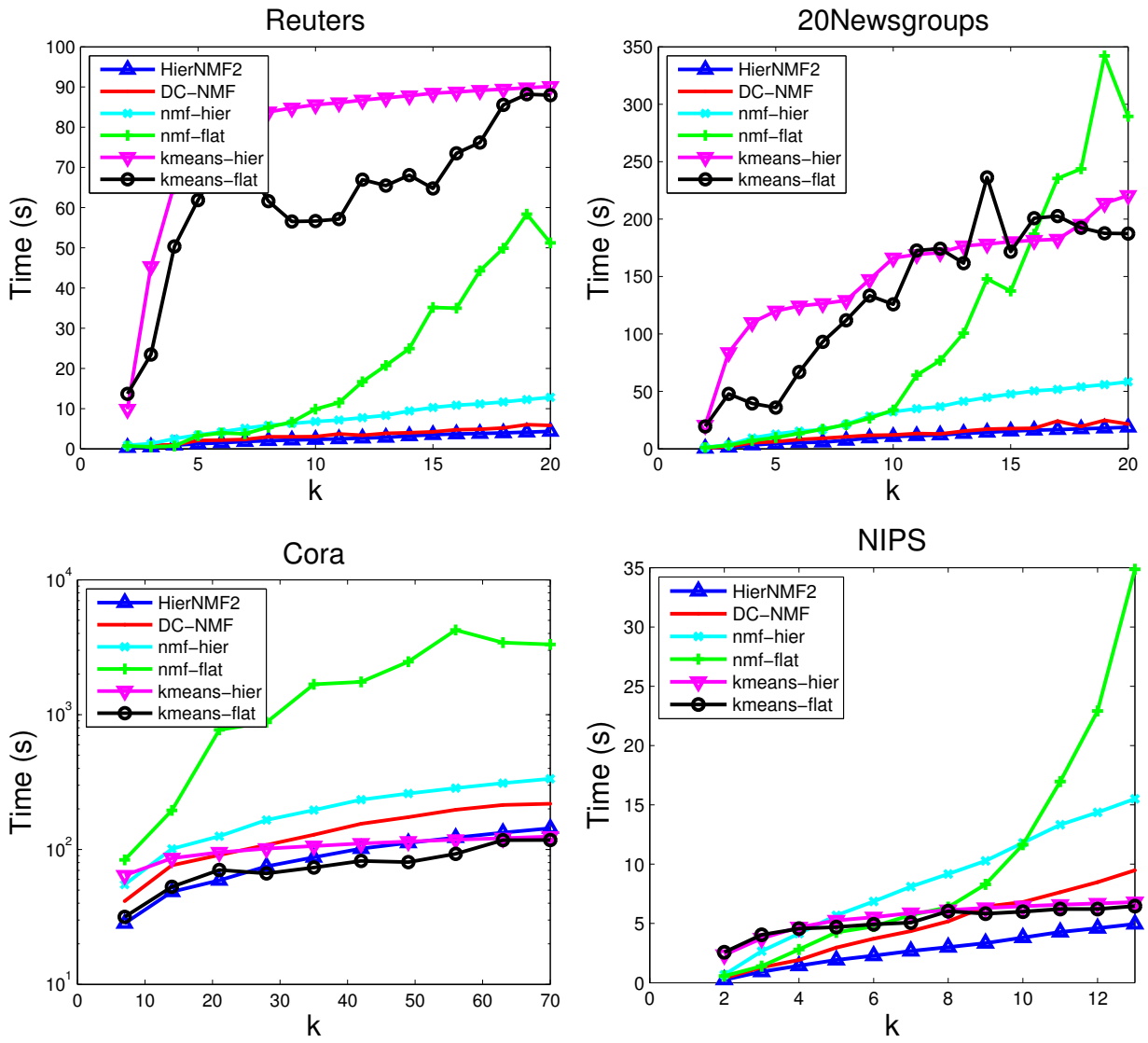


Figure 7: Timing results for the Matlab implementation of HierNMF2, DC-NMF, NMF, and K-means on the smaller data sets.

4.2.2 Timing Results

Fig. 7 shows the run-time of the proposed methods versus NMF and K-means, all implemented in Matlab. DC-NMF required substantially less run-time compared to the standard flat NMF. These results show that flat clustering based on standard NMF exhibits a superlinear trend while hierarchical clustering based on Rank-2 NMF exhibits a linear trend of runtime as k increases. For example, to generate 70 clusters on the **Cora** data set, HierNMF2, DC-NMF, `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 runtime to DC-NMF; however, the cluster quality is not as good, which was shown earlier in Fig. 5.

Fig. 8 compares the run-time of our C++ implementation of DC-NMF available in the software `smallk` [5] versus off-the-shelf toolkits (`CLUTO`, `Mallet-LDA`) and recent methods proposed for

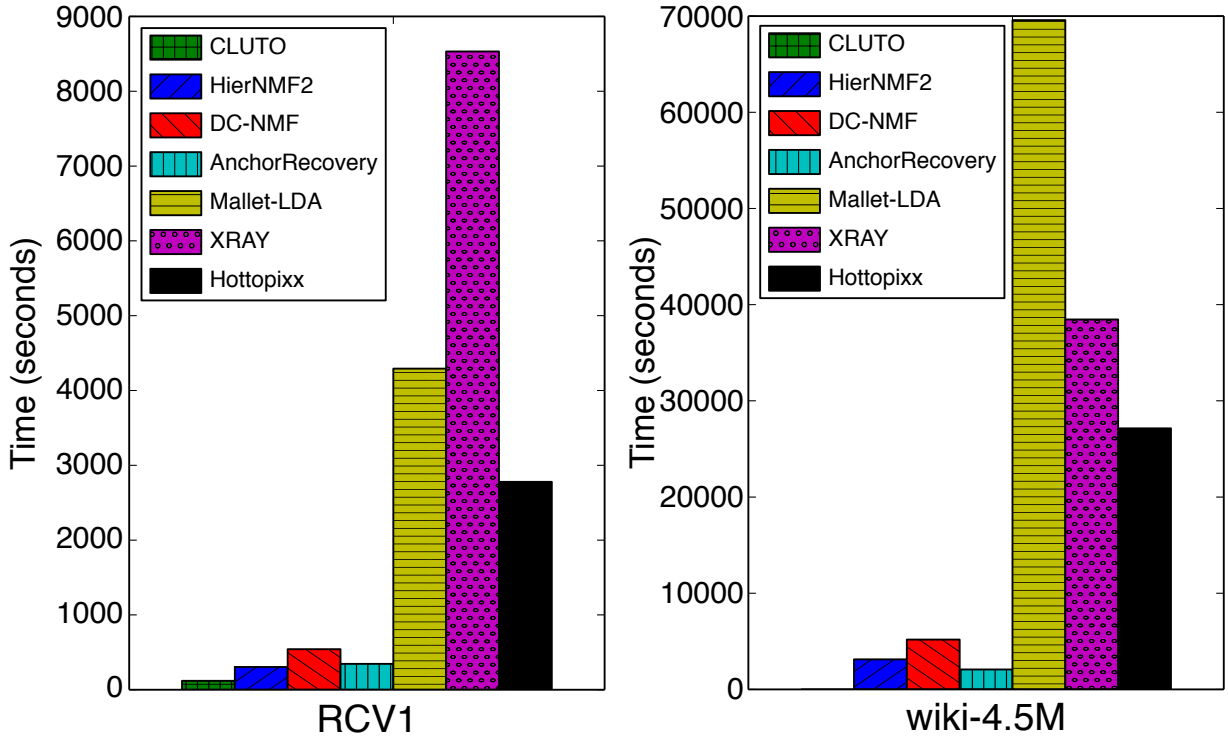


Figure 8: Timing results for the C++ implementation of HierNMF2 and DC-NMF available in our open-source software `smallk` and other state-of-the-art clustering and topic modeling methods on large, unlabeled text data sets.

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, DC-NMF is about 20 times faster than `Mallet-LDA`; particularly on the largest **Wiki-4.5M** data set in our experiments, DC-NMF 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 similarity matrix and thus have a large memory footprint [35, 36]. `AnchorRecovery`, on the other hand, computes a random projection of the word-word similarity matrix, greatly reducing the time and space complexity [34]; however, as we have seen in Fig. 6, its cluster quality is not as good as that of DC-NMF.

Overall, DC-NMF is the best-performing method in our experiments, considering both cluster quality and efficiency. The relatively recent software package `CLUTO` is also competitive.[10]

[10]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.

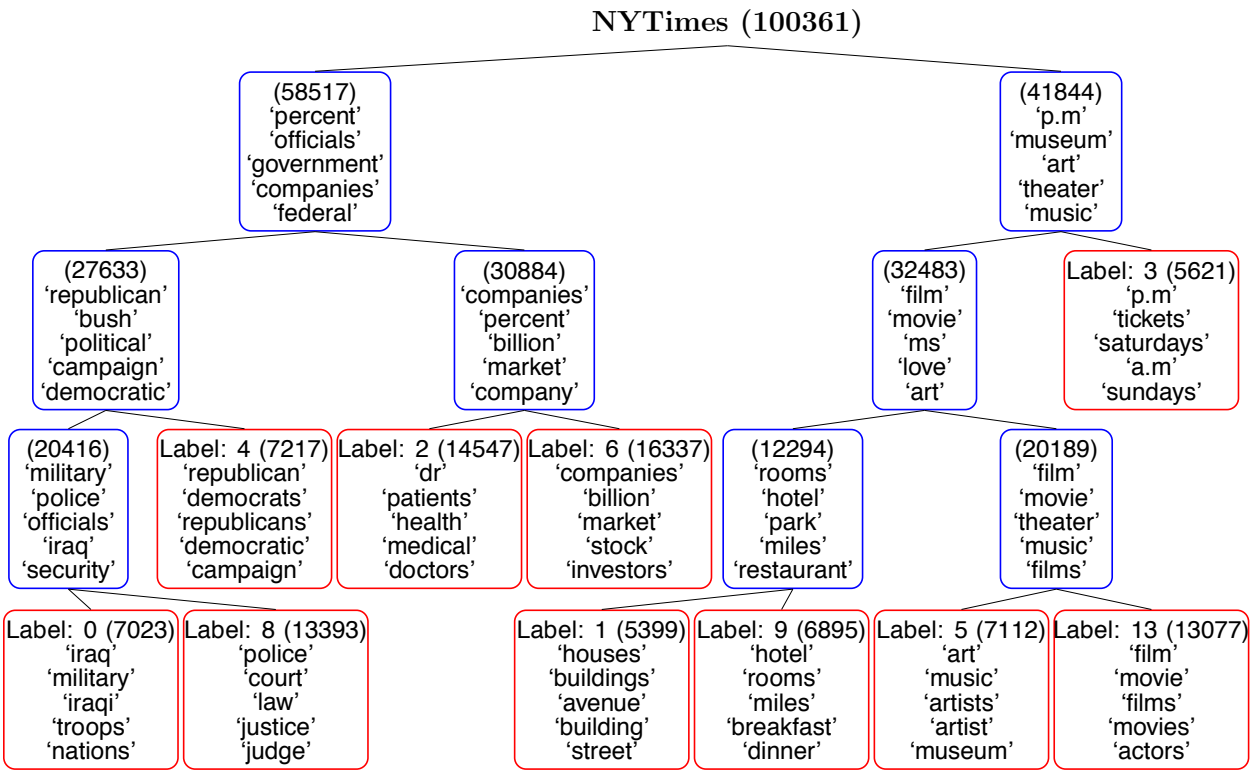


Figure 9: Hierarchical clustering result generated on a data set consisting of 100,361 New York Times articles for illustration. The hierarchy is automatically detected and not necessarily a balanced tree. Each tree node $\mathcal{N}$ is associated with a column of W, denoted as $\mathbf{w}_{\mathcal{N}}$, generated by Rank-2 NMF applied on its parent node. We display the five terms with highest importance values in $\mathbf{w}_{\mathcal{N}}$. 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.

To visualize the cluster/topic tree generated by HierNMF2, we show an illustration of the topic structure for a news article data set containing 100,361 articles in Fig. 9. First, we notice that the tree was not restrained to have a balanced structure, and HierNMF2 was able to determine the semantic organization on-the-fly. We can see that the articles were first divided into two big categories—politics/economy and art/entertainment/life. In the next few hierarchical levels, those topics (politics, economy, art, etc.) were further refined and emerged as more coherent sub-topics. Finally, at the leaf level, HierNMF2 produced fine-grained topics such as Iraq war, law and justice, stock market, movies, music, health, houses and hotels.

4.3 SymNMF Experiments

4.3.1 Methods for Comparison

We compare our algorithm with some recent algorithms. We use 8 threads for all methods that support multi-threading. For NISE (Neighborhood-Inflated Seed Expansion) [39] we are only able to use one thread because its parallel version exits with error in our experiments. For all the algorithms, default parameters are used if not specified. To better communicate the results, below

Table 3: Some statistics for ground truth communities from SNAP.

Data set	#Nodes	#Edges	Nodes that belong to 0 Community		Nodes that belong to 1 Community		
			Count	%	Count	%	Rel %
DBLP06	317080	1049866	56082	17.69%	150192	47.37%	57.55%
Youtube	1134890	2987624	1082215	95.36%	32613	2.87%	61.91%
Amazon	334863	925872	14915	4.45%	10604	3.17%	3.31%
LiveJournal	3997962	34681189	2850014	71.29%	394234	9.86%	34.34%
Friendster	65608366	1806067135	57663417	87.89%	3546017	5.40%	44.63%
Orkut	3072441	117185083	750142	24.42%	128094	4.17%	5.52%

The last few columns show the number of nodes that do not belong to any communities and the number of nodes that belong to only one community. The "Rel %" is the number of nodes that belong to one community divided by the number of nodes that belong to at least one community.

are the labels that denote each algorithm, which will be used in the following tables:

- `h2-n(g)(d)-a(x)`: These labels represent several versions of our algorithm. Here `h2` stands for HierSymNMF2, `n` for the `ncut_local` criterion, `ng` for the `ncut_global` criterion, and `ngd` for the `ncut_global_diff` criterion (see previous sections for the definitions of these criteria); '`a`' means that we compute the real normalized cut using the original **a**jacency matrix; and '`x`' indicates that an appro**x**imated normalized cut is computed using the normalized adjacency matrix, which usually results in faster computations. We stop our algorithm after $k-1$ binary splits where k is the number of communities to find. Theoretically, this will generate k communities. However, we remove fully disconnected communities, as outliers since they are often far from significant because of their unusually small sizes and they correspond to all-zero submatrices in the graph adjacency matrix, which does not have a meaningful rank-2 representation. Therefore, the final number of communities are usually slightly smaller than k, as will be shown in the "Experiment Results" section.
- `SCD`: SCD algorithm [40].
- `BigClam`: BigClam algorithm [41].
- `Graclus`: Graclus algorithm [42].
- `NISE`: An improved version of NISE that is published in 2016 [39].

4.3.2 Data sets

The data used for the experimental results of this paper are mostly from SNAP (Stanford Network Analysis Project) data sets [43, 44]. In our study, we found that the ground-truth information in SNAP is incomplete, for example, a large percentage of nodes does not belong to any ground-truth community. Table 3 shows some statistics regarding the number of communities to which each node belongs. Although all of these data sets can be conveniently accessed on the SNAP website as a graph with ground-truth communities, **DBLP06** is the only data set with a complete raw data set openly available to the public. The other five data sets (**Youtube**, **Amazon**, **LiveJournal**,

Friendster and **Orkut**) were obtained by crawling the web, and they are far from being complete. Crawling large complex graphs is challenging by itself that may need extensive and specialized research efforts. We do not aim to solve this issue in this paper. The **Orkut** and **Youtube** data sets can be acquired from [45]. Detailed descriptions are available explaining the crawling procedure and analysis of the completeness. It has been concluded that the **Orkut** and **Youtube** data sets are not complete. Such incompleteness in crawled data sets is expected due to intrinsic restrictions of web crawling such as rate limit and privacy protection. The **Friendster** data was crawled by the ArchiveTeam and the **LiveJournal** data comes from [46]. The **Amazon** data was crawled by the SNAP group [47]. However, information on how the data were collected and processed, and analysis of data completeness are not available.

Possible reasons that many nodes in these data sets do not belong to any communities are: (1) SNAP removed communities with less than three nodes, which caused some nodes to "lose" their memberships; (2) The well known incompleteness of crawled data sets; (3) For social networks (**Youtube**, **LiveJournal**, **Friendster**, and **Orkut**), it is common that a user does not join any user groups; (4) SNAP used the data set from [46] to generate the **DBLP06** data set, which was published in 2006. At that time, the DBLP database was not as mature and complete as it is today. Another issue of the above data sets is that all nodes are anonymized, which ensures protection of user privacy, but limits our ability to interpret community detection results.

The DBLP data is openly accessible, and is provided using a highly structured format—XML. We reconstructed the co-authorship network and ground-truth communities from a recent DBLP snapshot to obtain a more recent and complete DBLP data set with all of the meta information preserved (see the following subsection). Although the other data sets which we currently cannot improve are also valuable, our goal is to obtain new information from comparison of community detection results and ground truth communities, rather than simply recovering the ground truth communities.

4.3.3 Constructing the DBLP15 Data Set

DBLP is an online reference for bibliographic information on major computer science publications. [48]. As of June 17, 2015, DBLP has indexed 4,316 conferences, 1,417 journals and 1,573,969 authors [49]. The whole DBLP data set is provided in a well formatted XML file. The snapshot/release version of the data we use can be accessed at `http://dblp.dagstuhl.de/xml/release/dblp-2015-06-02.xml.gz`. The structure of this XML file is illustrated in Figure 10. The root element is the `dblp` element. We call the children of the root elements *Level 1 elements* and the children of Level 1 elements *Level 2 elements*, and so on. Level 1 elements represent the individual data records [50], such as `article` and `book`, etc. Since publication-venue relation makes more sense for the journal and conference papers, and these two types of publications occupy most of DBLP, we consider only `article` and `inproceedings` elements when constructing our data set. Level 2 elements contain the meta-information about the publications, such as title, authors, journal/proceeding names, etc.

Our goal is to obtain a co-authorship network and ground truth information (venue-author relation) from the XML file. Although the XML file is highly structured, such a task is still not straightforward due to the ambiguity of entities, such as conflicts or changes of author names, various abbreviations, or even journal name change. DBLP resolves the author ambiguity issue by using a unique number for each author. However, the venue ambiguity is still an issue in DBLP:

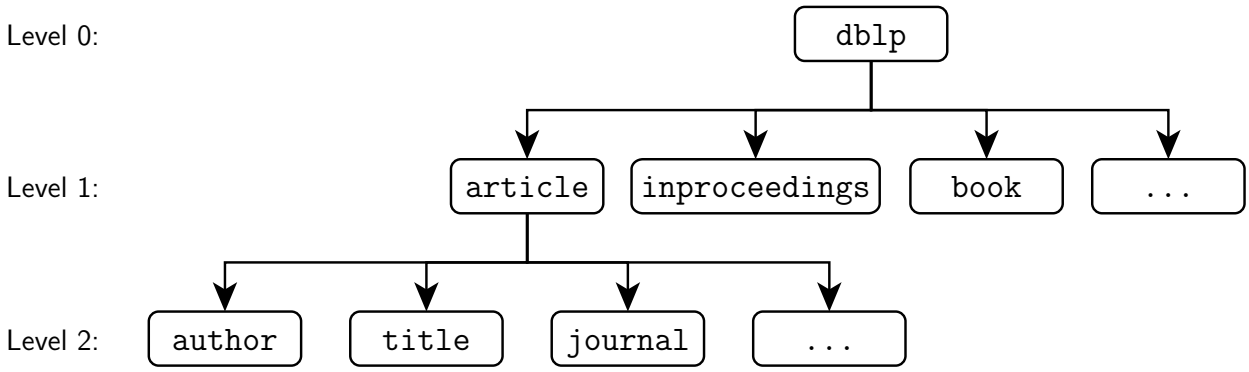
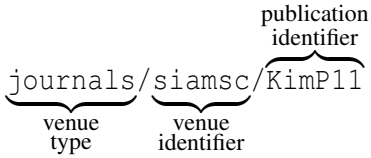


Figure 10: Structure of `dblp.xml`.

there are no unique identifiers for venues. Fortunately, each record in DBLP has a unique key and most paper' keys contain the venue information as follows:



However, there are still a few exceptions. To examine the validity of venue identifiers efficiently, we manually examine the identifiers not listed in the journal and conference index provided by the DBLP website, since such indices seem to be maintained by humans and assumed to be reliable. Using this process we found 5240 unique venues (journals or conferences).

Now unique identifiers for both authors and venues make extracting the network and community information very reasonable. The next step is to create a node for each author, and create a link between two authors if they have ever coauthored in the same publication. For community information, each venue is a community, and an author belongs to a community if he/she has published in the corresponding venue.

A few authors do not have any coauthor in the DBLP database, and become isolated nodes in the generated network. Thus, we remove these authors. However, after removing those authors, some venues/communities become empty because all of their authors are removed. So we remove those empty communities. After this cleaning, we obtained 1,509,944 authors in 5,147 communities (venues).

This cleaned network has 51,328 (weakly) connected components, where the largest connected component contains 1,357,781 nodes, which takes 89.9% of all nodes. The remaining 51,327 connected components are all small, the largest of which has only 37 nodes. We take the largest connected component as the network to study. By extracting the largest connected component, we obtain a network with 1,357,781 nodes, 6,369,212 edges and 5,146 ground truth communities. The ground truth communities were divided into connected components, obtaining 93,824 communities. The divided ground truth communities were used for comparison with detected communities.

The new **DBLP15** data set is available at `https://github.com/smallk/smallk_data/tree/master/dblp_ground_truth`.

4.4 JointNMF Experiment Results

We run our experiments on a server with two Intel E5-2620 processors, each having six cores, and 377 GB memory. The results are listed in Tables 4 to 11.

Table 4: Community detection results on DBLP06: internal measures

Algorithm	Number of Clusters	Coverage	Algorithm Time (s)	Total Time (s)	Average Ncut
h2-n-a	4982	98.57%	612.99	614.12	0.2089
h2-n-x	4981	98.55%	587.98	589.10	0.2174
h2-ng-a	4984	98.48%	921.99	923.14	0.1922
h2-ng-x	4982	98.50%	872.48	873.64	0.1921
h2-ngd-a	4986	98.64%	882.27	883.41	0.1767
h2-ngd-x	4984	98.66%	908.31	909.46	0.1774
SCD	139986	100.00%	1.89	4.52	0.8091
BigClam	5000	90.57%	N/A	230.59	0.6083
Graclus	5000	100.00%	161.70	162.01	0.2228
NISE	5463	99.33%	501.38	501.53	0.2026

Table 5: Community detection results on DBLP06: external measures

Algorithm	Number of Clusters	F1	Precision	Recall	Reverse Precision	Reverse Recall
h2-n-a	3312	0.4355	0.8804	0.5242	0.9005	0.4030
h2-n-x	3298	0.4236	0.8855	0.5071	0.9007	0.3937
h2-ng-a	3211	0.4417	0.8708	0.5492	0.8490	0.3996
h2-ng-x	3118	0.4374	0.8742	0.5497	0.8574	0.3898
h2-ngd-a	3192	0.4577	0.8575	0.5800	0.8719	0.4091
h2-ngd-x	3138	0.4534	0.8541	0.5808	0.8768	0.4008
SCD	34705	0.4644	0.9817	0.1268	0.7053	0.9755
BigClam	4952	0.3778	0.4857	0.6807	0.9269	0.3121
Graclus	4633	0.4765	0.6915	0.6006	0.8852	0.4517
NISE	4903	0.4118	0.5735	0.7942	0.9518	0.3552

In the "internal measures" table, "coverage" measures the percentage of nodes which are assigned to at least one community; "algorithm time" and "total time" provide the runtime information. We list two measures of runtime since our algorithm (and also NISE) implemented in MATLAB directly uses a processed matrix in memory as its input. Other algorithms must first read the graph stored as an edge list or an adjacency list and convert the graph to the appropriate internal representation. Therefore, we use "algorithm time" to measure the algorithm runtime without the time for reading and converting the graph, which is reported by the algorithms themselves. The "total time" is the wall clock time for running the algorithm, including the time for

Table 6: Community detection results on Amazon: internal measures

Algorithm	Number of Clusters	Coverage	Algorithm Time (s)	Total Time (s)	Average Ncut
h2-n-a	4989	98.84%	466.99	468.09	0.1657
h2-n-x	4988	98.80%	452.05	453.13	0.1711
h2-ng-a	4990	98.73%	537.82	538.91	0.1617
h2-ng-x	4988	98.66%	514.71	515.81	0.1709
h2-ngd-a	4990	98.82%	573.64	574.73	0.1491
h2-ngd-x	4990	98.79%	560.86	561.96	0.1545
SCD	141405	100.00%	1.86	4.37	0.8418
BigClam	5000	97.31%	N/A	169.51	0.3198
Graclus	5000	100.00%	119.25	119.45	0.1450
NISE	5182	99.63%	990.84	990.86	0.1118

Table 7: Community detection results on Amazon: external measures

Algorithm	Number of Clusters	F1	Precision	Recall	Reverse Precision	Reverse Recall
h2-n-a	1069	0.7883	0.9747	0.8179	0.9057	0.7593
h2-n-x	1038	0.7717	0.9787	0.8109	0.9070	0.7311
h2-ng-a	1209	0.7422	0.9657	0.7247	0.8748	0.7622
h2-ng-x	1185	0.7268	0.9655	0.7152	0.8743	0.7372
h2-ngd-a	1181	0.7813	0.9698	0.7741	0.8867	0.7922
h2-ngd-x	1168	0.7725	0.9702	0.7681	0.8869	0.7792
SCD	3841	0.6202	0.9998	0.3166	0.8186	0.9948
BigClam	1447	0.8389	0.9718	0.7824	0.9574	0.8744
Graclus	991	0.8555	0.9356	0.9471	0.9892	0.7525
NISE	2612	0.6673	0.6666	0.9733	0.9807	0.5390

Table 8: Community detection results on Youtube: internal measures

Algorithm	Number of Clusters	Coverage	Algorithm Time (s)	Total Time (s)	Average Ncut
h2-n-a	3782	98.10%	1182.39	1185.94	0.1681
h2-n-x	3780	98.01%	1189.09	1192.66	0.1634
h2-ng-a	3798	98.00%	1885.15	1888.71	0.1520
h2-ng-x	3851	98.14%	1816.98	1820.45	0.1491
h2-ngd-a	3886	98.27%	1613.13	1616.57	0.1395
h2-ngd-x	3874	98.22%	1621.04	1624.50	0.1428
SCD	998722	100.00%	12.03	20.39	0.9882
BigClam	5000	41.51%	N/A	2379.84	0.7398
Graclus	5000	100.00%	2160.11	2168.36	0.4919
NISE	5162	99.96%	2598.25	2598.66	0.4313

Table 9: Community detection results on Youtube: external measures

Algorithm	Number of Clusters	F1	Precision	Recall	Reverse Precision	Reverse Recall
h2-n-a	189	0.2907	0.9639	0.5247	0.9810	0.0403
h2-n-x	193	0.2972	0.9645	0.5411	0.9790	0.0412
h2-ng-a	241	0.2935	0.8684	0.5969	0.9315	0.0516
h2-ng-x	259	0.3027	0.8932	0.5915	0.9467	0.0551
h2-ngd-a	227	0.3030	0.9299	0.5694	0.9594	0.0484
h2-ngd-x	238	0.2978	0.9394	0.5476	0.9633	0.0507
SCD	27864	0.3652	0.9709	0.1330	0.4453	0.9841
BigClam	3850	0.2354	0.3755	0.5187	0.4743	0.2370
Graclus	3802	0.3827	0.5761	0.5348	0.6532	0.4148
NISE	3778	0.2720	0.4762	0.7180	0.9912	0.2580

Table 10: Community detection results on DBLP15: internal measures

Algorithm	Number of Clusters	Coverage	Algorithm Time (s)	Total Time (s)	Average Ncut
h2-n-a	4982	99.66%	1648.73	1654.71	0.1702
h2-n-x	4982	99.67%	1666.13	1672.01	0.1743
h2-ng-a	4984	99.62%	3262.76	3268.63	0.1606
h2-ng-x	4984	99.64%	3220.70	3226.57	0.1568
h2-ngd-a	4987	99.69%	2558.80	2564.60	0.1457
h2-ngd-x	4987	99.70%	2503.58	2509.38	0.1463
SCD	565235	100.00%	16.89	33.22	0.8357
BigClam	5000	65.07%	N/A	1352.57	0.6761
Graclus	5000	100.00%	1980.38	1987.97	0.2732
NISE	5101	86.77%	945.15	945.90	0.3482

Table 11: Community detection results on DBLP15: external measures

Algorithm	Number of Clusters	F1	Precision	Recall	Reverse Precision	Reverse Recall
h2-n-a	4982	0.3028	0.7282	0.7000	0.9830	0.0445
h2-n-x	4982	0.2994	0.7229	0.6986	0.9833	0.0442
h2-ng-a	4984	0.3025	0.7188	0.7164	0.9066	0.0440
h2-ng-x	4984	0.2992	0.6978	0.7275	0.9095	0.0439
h2-ngd-a	4987	0.3036	0.6963	0.7455	0.9640	0.0446
h2-ngd-x	4987	0.3016	0.6839	0.7512	0.9658	0.0446
SCD	565235	0.3477	0.8684	0.1050	0.5803	0.8218
BigClam	5000	0.0784	0.2357	0.9875	0.6806	0.0192
Graclus	5000	0.0861	0.2411	0.9874	0.7576	0.0275
NISE	5101	0.0955	0.3606	0.8307	0.7066	0.0253

reading and converting the graph, which is measured with an external timer. BigClam reports its algorithm time as the sum of time used in each core and therefore the results are not comparable. For completeness, we added the data loading and preprocessing time, which is measured separately, to obtain a "total time" for MATLAB algorithms (our algorithm and NISE).

The "number of clusters" in the "internal measures" table is different across different methods due to the following reasons. The SCD algorithm does not provide an interface for specifying the number of communities to detect, and instead detects the number of communities automatically. For other algorithms, we specify the number of communities to detect as 5000. The actual number of communities generated by HierSymNMF2 is usually smaller than 5000, as discussed in the "Methods for Comparison" section. Also, the number of communities generated by NISE are usually a little larger than 5000, which is also an expected behavior [39].

In the "external measures" table, the "reverse precision" and "reverse recall" refer to the scores computed as if the ground truth communities are treated as detected communities and the detected communities are treated as the ground truth, respectively. Note that the number of clusters in "external measures" is smaller than the one in "internal measures" due to the removal of nodes that do not appear in the ground truth.

We have the following observations from the experimental results: (1) Our HierSymNMF2 algorithm has significant advantages over other methods in average normalized cut on most data sets except the **Amazon** data set. On the **Amazon** data set, HierSymNMF2 achieves much lower average normalized cut than SCD and BigClam, and the variant h2-ngd-a obtained comparable average normalized cut (0.1491) versus Graclus (0.1450), which is not as good as NISE (0.1118) though. (2) HierSymNMF2 runs slower than most other algorithms on **DBLP06** and **DBLP15**. On the **Youtube** data set, HierSymNMF2 runs faster than BigClam, Graclus and NISE. On the **Amazon** data set, HierSymNMF2 runs faster than NISE, but slower than other methods. (3) HierSymNMF2 achieves better F1 score than BigClam and NISE on all the data sets we used. Graclus has better F1 score than HierSymNMF2 on **DBLP06**, **Amazon**, **Youtube** data sets but obtained an unusually low F1 score on the **DBLP15** data set. SCD achieves higher F1 scores than HierSymNMF2. However, SCD often discovers a significantly larger number of (non-overlapping) communities than expected and has very unbalanced precision and recall scores compared to other algorithms. The SCD algorithm finds a number of communities as it finds the communities and the number of communities cannot be given to SCD as an input. The SCD algorithm starts by assigning an initial partitioning of the graph heuristically. In short, in the initial partitioning, each node and all its neighbors form a community, and special care is taken to ensure that no node belongs to more than one community. As a result, this initial step often creates a lot more number of communities than the optimal number, though later refining procedures may reduce the number of communities. As can be seen from the experiment results, when compared to BigClam, Graclus, NISE and our proposed algorithms that take the number of communities as an input, a much larger number of communities that the SCD generates does not necessarily translate to a better overall community detection result in terms of either normalized cut or F1 scores.

4.4.1 Clustering US Patent, BlogCatalog and Flickr Data

All experiments were performed on a server with two Intel(R) Xeon(R) CPU E5-2680 v3 CPUs and 377GB memory.

The main data set used for the experiments is the US patent claim and citation data from

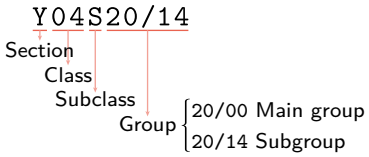


Figure 11: An example classification label in the CPC scheme

PatentsView[11]. Some advantages of using US patents as a data source are: (1) the openness, centralized management and availability of relatively structured data format makes the patent data easier to obtain and process; (2) the abundance of the patent database ensures enough samples that can be studied; (3) patents were carefully assigned with classification labels, and such labels were examined by patent examiners; therefore the classification information can be used as a relatively reliable ground truth.

We use the Cooperative Patent Classification (CPC) system, where each classification label has the scheme illustrated in Figure 11. We select 13 CPC classes (A22, A42, B06, B09, B68, C06, C13, C14, C40, D02, D10, F22, Y04) and use patents under each class to construct 13 different data sets[12]. For each data set, we first construct the term-document matrix representing the patent claims and the graph adjacency matrix representing the patent citation relations. Our algorithm requires a symmetric adjacency matrix and therefore we treat the citation graph as undirected by ignoring the directions. We then clean the data by removing terms that appear very infrequently and documents that are too short or duplicated, and extract the largest connected components of the graph. Finally, we apply tf-idf to the term-document matrix, normalize its columns to have unit 2-norm, obtaining the matrix X, and let S be $D^{-1/2}AD^{-1/2}$, where $A \in \mathbb{R}^{n\times n}$ is the adjacency matrix, $D = \mathrm{diag}(d_1,\ldots,d_n)$ and $d_i = \sum_{j=1}^{n} A_{ij}$ is the degree of vertex i. We use CPC groups as ground truth clusters. Some statistics about these data sets (after cleaning) are listed in Table 12.

To verify our algorithm on other types of data, we also use the BlogCatalog data set from [51] and the Flickr data set from [52]. These data sets have users as graph nodes and represent user commenting and friendship relations as graph edges. The content comes from user generated keywords/tags that are used to describe their blog articles (BlogCatalog) or photos (Flickr), which is different from traditional text content. The ground truth clusters of BlogCatalog data set are defined by categories of each blog and the ones for the Flickr data set are defined by user groups. We apply the same preprocessing as for the US patent data sets. Some statistics regarding these two data sets (after preprocessing) are listed in Table 13.

Since the ground truth clusters have overlapping, we use average F1 score and Rand index, as the measures for the evaluation of the clustering results.

We compare our algorithm with NMF and SymNMF, which have leading performance in text clustering and graph clustering, respectively. For hybrid clustering, we choose PCL-DC [53] to compare with based on its popularity and source code availability. While our method is based on nonnegative matrix factorization, PCL-DC is a probabilistic method that combines a conditional model for link analysis and a discriminative model for content analysis. Although we mentioned many other algorithms, we found that for other algorithms, either the code is not available or the code is available but we encountered runtime errors during experimental tests. Both JointNMF and

[11] `http://www.patentsview.org`

[12] These data sets are available at `http://smallk.github.io/pages_about.html`

Table 12: Some statistics of US patent data sets.

Class	#Patents	#Citations	#Groups
A22	4976	28746	230
A42	4213	29285	134
B06	2938	11549	82
B09	3522	17302	38
B68	790	2433	93
C06	3347	17562	141
C13	1010	3717	87
C14	583	1125	69
C40	3748	28854	41
D02	3170	11216	158
D10	2548	8486	154
F22	3040	7977	359
Y04	3242	21518	76

Table 13: Some statistics of BlogCatalog and Flickr data sets.

Data	#Nodes	#Edges	#Tags	#Groud truth clusters
BlogCatalog	31228	782584	5387	60
Flickr	32576	2749800	77234	170

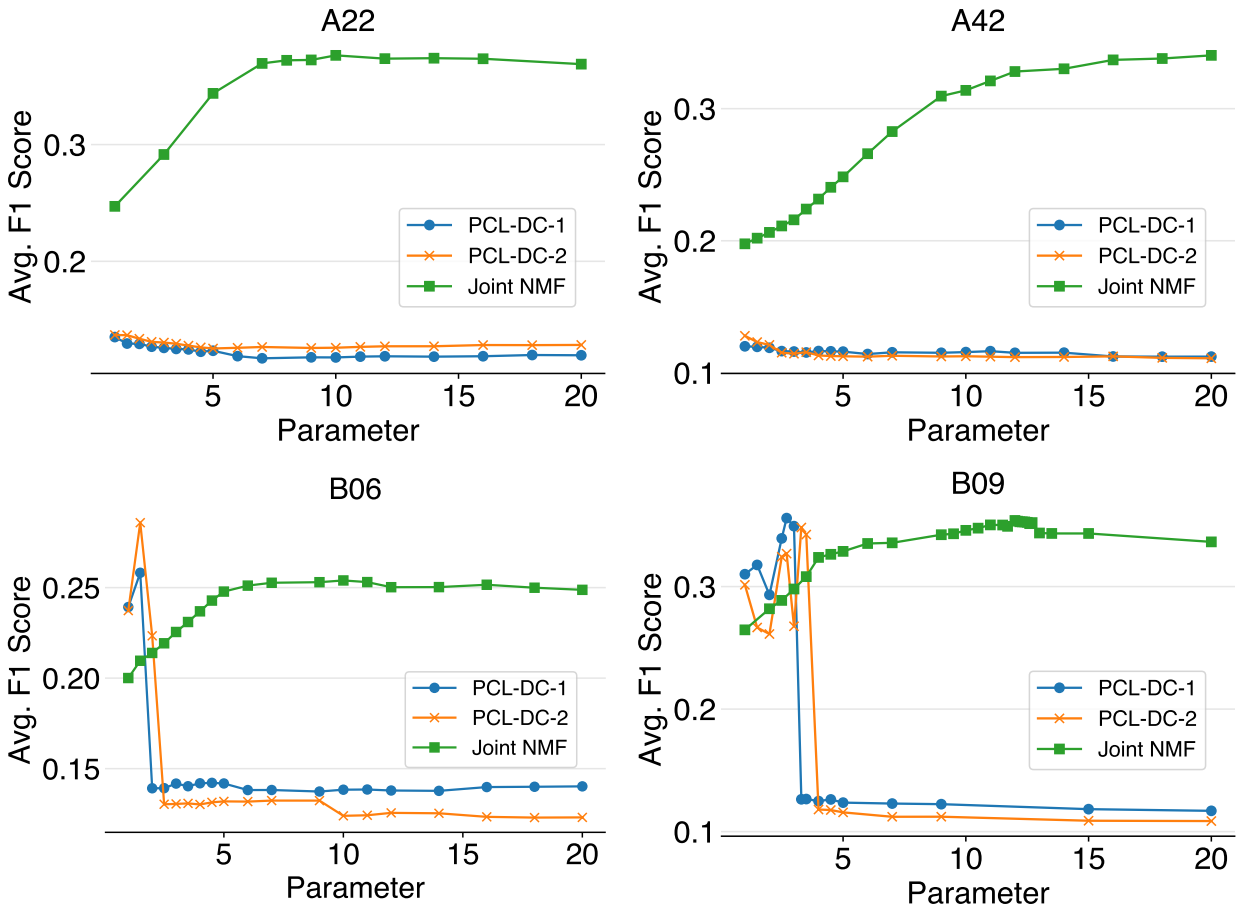


Figure 12: Parameter sensitivity of PCL-DC and JointNMF. The parameter of PCL-DC is λ and the parameter of JointNMF is α.

PCL-DC have parameters to set. For JointNMF, we let the default parameter be $\alpha = \|X\|_F^2/\|S\|_F^2$, meaning half-half balance between graph clustering and text clustering, and set $\beta = \alpha\|S\|_{max}$, where $\|S\|_{max}$ is the maximum absolute value of elements in S. The authors of PCL-DC do not provide a method to specify its regularization parameter λ. Therefore, it is important to first study how the parameter change will affect the algorithm performance. It is found that for $\lambda < 1$, PCL-DC sometimes becomes extremely slow, such that it may take weeks to run over all the data sets (estimated based on sampling run). Therefore, λ is varied within $[1,20]$. In Figure 12, we show how the average F1 score changes when λ varies in that range for the first four data sets listed in Table 12. The code of PCL-DC[13] provides two models (popularity link model and productivity link model), which we label as PCL-DC-1 and PCL-DC-2, respectively. The performance change of JointNMF when its parameter α varies in the same range is also studied. We observe that the PCL-DC is either worse than JointNMF or very sensitive to the parameters, and it is concluded that when λ exceeds a certain threshold (depending on the data), there is a large drop in clustering quality. Therefore, to have a tolerable run time while having a fair clustering quality, $\lambda = 1$ is chosen for the comparison experiments. The results of the comparison are listed in Table 14 to

[13] https://homepage.cs.uiowa.edu/~tyng/codes/community_detection.zip

Table 14: Hybrid clustering results: comparison of average F1 scores

Class	JointNMF	NMF	SymNMF	PCL-DC-1	PCL-DC-2
A22	**0.3730**	0.2293	0.3457	0.1351	0.1369
A42	**0.3215**	0.1779	0.3199	0.1201	0.1280
B06	**0.2502**	0.1905	0.2307	0.2393	0.2373
B09	**0.3336**	0.2449	0.2690	0.3101	0.3014
B68	0.3806	0.3044	0.3730	**0.4034**	0.3671
C06	**0.2257**	0.1830	0.2004	0.1156	0.1158
C13	**0.2990**	0.2664	0.2953	0.2616	0.2224
C14	0.3584	0.3232	**0.3603**	0.2692	0.2659
C40	0.1939	0.1709	0.1673	0.1951	**0.1981**
D02	**0.2990**	0.2131	0.2683	0.1756	0.2268
D10	**0.3046**	0.2452	0.2783	0.1612	0.2999
F22	**0.3006**	0.2211	0.2926	0.1533	0.1388
Y04	0.2489	0.2029	0.2019	**0.2599**	0.2596
blogcatalog	0.2038	0.2150	0.0750	**0.2754**	0.2754
flickr	0.1545	0.0748	**0.1660**	0.0855	0.0855

Table 16, where each value is the average over 10 runs.

Using these patent data sets, from our experiments it can be observed that: (1) JointNMF usually has the best average F1 scores, and its average F1 score is almost always better than that of NMF or SymNMF alone; (2) JointNMF and SymNMF have the best rand index; (3) SymNMF is usually the fasted algorithm; (4) The run time varies in a very different pattern between NMF based methods and PCL-DC. The algorithms for both NMF based methods and PCL-DC are iterative. For NMF based methods, the run time of each iteration is linear with respect to data size (e.g. number of nodes and edges) and cubic with respect to the number of clusters [10]. For PCL-DC, the run time of each iteration is linear with respect to both data size and the number of clusters [53]. If we compare Table 16 with Table 12, we can observe that for NMF methods the number of clusters does dominate the run time but for PCL-DC the run time is rather unpredictable, which may suggest that the convergence behavior of PCL-DC is not consistent over different data sets. On BlogCatalog and Flickr data sets, which have different kinds of content and graph edges, the performance varies depending on the data. However, the performance of JointNMF is comparable to the best method with the exception of run time on the Flickr data set.

In conclusion, for patent data sets, based on content and citations, JointNMF produces better quality solutions for clustering; for prediction of pairwise connection, both JointNMF and SymNMF perform well; speed-wise, JointNMF is not the fastest, but is comparable to other methods. On other types of data, the performance of each method varies, and JointNMF generates comparable results. The JointNMF method has other advantages: its parameter has explicit meanings (weight between text and graph), the clustering quality is not very sensitive to the parameter setting, and its default parameter works very well.

Table 15: Hybrid clustering results: comparison of rand index

Class	JointNMF	NMF	SymNMF	PCL-DC-1	PCL-DC-2
A22	**0.9785**	0.9768	0.9772	0.9274	0.9489
A42	**0.9650**	0.9633	0.9647	0.9225	0.9318
B06	**0.9368**	0.9357	0.9024	0.8775	0.8815
B09	**0.8497**	0.8387	0.7600	0.8464	0.8333
B68	0.9496	0.9423	**0.9508**	0.9272	0.8897
C06	0.9175	0.9150	**0.9182**	0.8969	0.8967
C13	0.8918	0.8873	**0.8927**	0.8598	0.8485
C14	**0.9086**	0.9036	0.9071	0.8233	0.7934
C40	0.6575	0.6507	**0.6820**	0.6593	0.6692
D02	**0.9612**	0.9594	0.9578	0.8922	0.8831
D10	**0.9080**	0.9048	0.9075	0.8676	0.8771
F22	0.9811	0.9797	**0.9816**	0.9554	0.9549
Y04	**0.8879**	0.8853	0.8697	0.8668	0.8622
blogcatalog	0.7572	**0.7652**	0.6173	0.7259	0.7259
flickr	0.0560	0.0409	**0.0782**	0.0620	0.0620

Table 16: Hybrid clustering results: comparison of run time (seconds)

Class	JointNMF	NMF	SymNMF	PCL-DC-1	PCL-DC-2
A22	769.4	304.4	219.2	**55.6**	57.5
A42	311.9	161.9	163.1	**24.3**	24.8
B06	193.8	115.8	**59.8**	444.5	1800.8
B09	145.6	109.6	**48.2**	406.6	588.8
B68	48.2	60.8	**7.6**	288.3	439.0
C06	489.8	269.0	160.6	21.1	**20.9**
C13	70.9	76.1	**8.8**	421.5	377.2
C14	29.5	25.0	**4.7**	220.6	83.8
C40	240.8	127.8	**54.3**	394.0	597.3
D02	534.5	238.5	**117.3**	1623.5	831.8
D10	280.8	155.4	95.9	**14.7**	1728.4
F22	1294.1	404.4	267.2	38.4	**36.7**
Y04	291.9	125.8	**103.8**	1568.3	987.6
blogcatalog	401.3	**222.8**	1515.6	4463.4	4522.4
flickr	12455.6	2437.9	3504.5	**1181.3**	1236.0

Table 17: Case study on Enron email data: frequency of number of memberships

#memberships	1	2	3	4	5	6	7	11
#employees	1069	149	45	17	8	7	1	1

4.4.2 Activity and Leader Detection from Enron Email Data

In an organization where various groups of people work on different subjects and engage in different activities, JointNMF can be used to detect such group structure, reveal the working subject/activities and find administrators/leaders in the organization. We assume that (1) within-group communications (e.g. emails) reflects the subject on which the team is working/activities engaged in and (2) people involved in multiple groups would likely hold a higher position in the organization, since they may be in charge of these groups. Each communication can be seen as a hypergraph edge that connects all people involved in the communication and the communication content is the text associated with the edge. Clustering the text data can distinguish and identify different working subjects/activities and clustering the graph data can divide people into workgroups. JointNMF utilizes both types of data simultaneously and therefore can distinguish different groups of people working on the same subject and different subjects worked on by the same group of people. After clustering, one can count and compare the number of groups/clusters each person belongs to—the more groups a person belongs to, the more likely the person is in a leadership or administrative position.

A subset of Enron email data extracted by a group from UC Berkeley [14], containing 1702 emails is used. First we construct the term document matrix from email content and the hypergraph incidence matrix from email-sender/recipient relations. The hypergraph has Enron employees as vertices and their emails as edges, and a vertex is connected by an edge if and only if the corresponding employee is the sender or a recipient of the corresponding email. After that, we clean the data by removing terms that appear very infrequently and emails that are too short or duplicated, and extracting the largest connected components of the hypergraph. The tf-idf transformation is then applied to the term-document matrix, its columns are normalized to have unit 2-norm, which obtains the matrix X. Finally, we apply JointNMF with $\alpha = \|X\|_F^2/\|S\|_F^2$ and $\beta = \alpha\|S\|_{max}$ to find 20 groups of employees. Note that since the dual hypergraph is used, the resulting clusters are clusters of emails rather than clusters of employees. To induce clusters of employees, one simply inserts employees involved in the same cluster of emails into one employee cluster. In this way, we can actually induce overlapping employee clusters from non-overlapping email clusters. It is assumed that an employee has j memberships if the employee belongs to j clusters. The number of memberships is counted for each employee and the frequency of each number is listed in Table 17. Employees that had at least 6 memberships are examined in online news and we found that they all held relatively high positions in Enron. Their names and positions are listed in Table 18. To see the effect of our algorithm on topic modeling, we list some topic keywords for each cluster in Table 19. It can be observed that some emails are communications about/with other companies and regulatory agencies (0,3,19); some are about administrative tasks or daily work (5,7,8,13,15,16,18); some are about legal issues (6,10); and some are related to the California energy crisis (2,11).

[14] http://bailando.sims.berkeley.edu/enron_email.html

Table 18: Case study on Enron email data: employees that has j memberships ($j \geq 6$) and their positions in Enron

j	Name	Position in Enron
11	Steven Kean	Chief of staff
7	Jeff Dasovich	Governmental affairs executive
6	Susan Mara	California director of Regulatory Affairs
	Richard Shapiro	VP of regulatory affairs
	Paul Kaufman	VP of Government Affairs
	James Steffes	VP of Government Affairs
	Tim Belden	Head of trading
	Richard Sanders	VP of Enron Whole Sale Services
	Joe Hartsoe	VP of Federal Regulatory Affairs

VP: vice president

Table 19: Case study on Enron email data: topic keywords of clusters

#	Keywords
0	ubs, warburg, forecast, confidential, win
1	blackberry, handheld, wireless
2	california, power, confidential, tariff, pursuant
3	caiso, refund, ferc, proceedings
4	burrito, peace, things, price, market, board, california
5	document, fax, tonight, sign, back, attach, thanks
6	wholesale, policy, compliance, receipt, legal, service
7	enron, please, know, attach, meeting, contact, call, any, time
8	london, conference, meeting, next, week
9	handheld, blackberry, wireless, agreement, confidential
10	testify, witness, fault, burden, cut, budget
11	california, electricity, energy, price, market, power, rate, bill
12	recommendation, template, participant, management
13	passcode, please, effective, confidential, change
14	stanford, university, expert, try, best, mail, california
15	account, invoice, trust, fund, transfer
16	expense, report, employee, name , approve, amount
17	folder, info, audit, access, apollo, email, sensitivity, server
18	sent, talk, presentation, thanks, infrastructure, amendment
19	hpl, aep, agreement, compete, deal, arrangement

4.5 Summer Challenge and Hackathon Results

4.5.1 Summer Challenge NBA Data

Correlated NMF (CoNMF): Matrix factorization methods, especially with non-negative constraints, gained its popularity due to its capability to produce quality topics fast. The topic discovery in this context is solely for input features, and labels are usually just fixed values. There are some approaches that jointly discover topics or mapping from multiple sources. A critical point here is that the rich features from different sources indeed subsumes similar characteristics. In this context, we consider a NBA game with game event statistics and its commentaries. If the NBA game was 'interesting', the statistics of game will be slightly different from others and so as comments. We studied correlation between multiple data sources to consider the inter-relationship. Specifically, we use correlation between embeddings of two using its own topics. For example, we can consider a latent concept that will change game event statistics and its comments in the similar way. We jointly minimize the matrix reconstruction loss on both data sources and maximize the correlation between two embedding.

We have defined the joint loss and implemented one dimensional case. The results discovered a hidden concept: interestingness. Changes of game statistics topic seems to towards to penalize ordinary game plays (normal jump shots), but weights more on game changer events: such as substitute player, or serious fouls. Similarly in comments, vocabularies that are relevant to express the interestingness are more weighted: '!', '?', 'lol', 'win', and 'good'.

Fast Spammer Detection Using Structural Rank (see http://arxiv.org/abs/1407.7072 for details) Comments of the NBA game dataset were infested with huge number of spammers. We developed a new spammer detection technique based on their behavior. Most of their contents are repeated under the same author name. There are several other methods that compute content similarity to detect such spammers. They make use of language model, or set intersection similarity, or average cosine similarity. We, instead, use a much faster but an effective method: the structural rank of author specific term-document matrix to detect the spammers. We used structural rank for computing content similarity of a set of documents. 1) Solely considering the non-zero pattern will be enough to measure the content similarity of a set of documents. 2) Term-document matrices are usually very sparse and our case is even sparser as we deal with very short documents (comments). Bipartite graph traverse algorithm will be extremely efficient in this case. 3) It is also much faster than other pairwise based similarity metrics. Our result showed that the proposed method is far faster than other methods (numeric rank, sparse numeric rank, or average pairwise cosine similarity) in large magnitude. We believe this efficient computation is especially useful in practice since we usually have billions of users and comments.

4.5.2 Summer Challenge WDC data set

Spamming Website Detection Based on NMF: We designed a method for detecting spamming sites based on the leading eigenvector of the adjacency matrix A. The method iteratively removes the spamming cliques from the graph and considers the leading eigenvector of the remaining adjacency matrix. Furthermore, we designed methods to detect clusters existing in the hyperlink graph. Specifically, we treat the co-citation matrix A^TA as a similarity matrix and expect to obtain the authority clusters, and then treat the bibliographic coupling matrix AA^T as similarity matrix

and expect to obtain the "hub clusters". We applied SymNMF and Rank2 NMF to A^TA and AA^T for better interpretation of the authority and hub clusters. Postprocessing clustering is not needed.
Text-Based Link Prediction Using Joint NMF: In our attempt of getting insight of what domain/page names say about linkages, we found that domain names are often chaotic but page names usually provide valuable information about a web page. We are developing machine learning algorithms based on nonnegative matrix factorization to predict possible hyperlinks given page names.

By joining standard NMF with symmetric NMF, our method is able to find low-rank representations of web pages that can reflect both page contents (titles) and linkage information. With these information learned, our method can predict possible links for new text input.

We have developed fast algorithms for solving our proposed joint NMF problem based on a 3-block coordinate descent method. We've applied our method to a test data set, and it turns out that our method can achieve an almost zero false positive rate on the test set, which means the link we predicted are very likely to be a real link.

4.5.3 Summer Challenge Akamai CIDR data set

We propose a new method of pattern analysis and outlier detection for Internet traffic data sets. In particular, we study the CIDR data set that records the traffic of categorical content through IP blocks from all over the world. A preprocessing technique is introduced to construct the features that reflect the daily activities of the observation points. A low-rank model based on $L_{2,1}$ robust nonnegative matrix factorization (NMF) is proposed to deal with the temporal redundancy among the daily Internet traffic and the sparsity of anomalies. Moreover, a multi-level model is built upon the low-rank model to incorporate the geospatial information of observation points. Experimental results show that the low-rank model is very useful to extract the traffic patterns of individual observation points as well as the traffic patterns that summarize observation points from a certain region. The model also finds interesting unusual behaviors from a few observation points that can be connected to real-world events.

4.5.4 Summer Challenge Kiva data set

We have published two refereed conference papers based on the methods and data analyis results on Kiva data set [54, 55].

4.5.5 January 2016 Hackathon on Building Permit Datasets

We applied our topic modeling software (available in SmallK) on relevant fields from the permit datasets provided, filtered by location and time as appropriate to search for the impact of local weather events.

4.5.6 May 2016 Hackathon on Yemen Ceasefire Violation

We applied our topic modeling software (available in SmallK) on relevant fields from the Yemen datasets provided, filtered by location and time as appropriate to search for ceasefire violations. We focused most of our attention on the Telegram dataset. After the initial topics were obtained, we identified topics of interest that appeared related to ceasefire with the assistance of a Subject

Matter Expert. We then compared underrepresented documents with the identified related topics to discover more related topics and remove irrelevant documents. Such comparison is possible because in NMF-based topic modeling methods, topics and documents are encoded in the same vector space.

We also analyzed the Instagram datasets, focusing on the network information contained therein. The "comments" and "like" feature in Instagram implies a network structure: user A is connected to user B if user A has liked or commented user B's post. We analyzed the influencers in the network by HITS algorithm and detected communities in the network by HierSymNMF2 algorithm. It turned out that most top influencers were professional photographers and their photos were about food, landscape, etc. We didn't find much content related to ceasefire violations specifically.

4.5.7 September 2016 Hackathon on Patent Data

We gained some insights about the patent data by applying our NMF based clustering algorithms. We used the patent abstracts as the text data, and constructed a similarity graph using CPC classification—two patents have an edge if they share the same classification. Then we applied our late-fusion hierarchical clustering on the hybrid data (with texts and the classification graph). We first apply HierSymNMF2 to cluster the graph, and then on each hierarchy node, we apply HierNMF2 to generate five topics. We observed that the graph clusters had unbalanced sizes and those large clusters were typically about recent technology such as wireless network and interactive media while smaller clusters usually corresponded to mature/traditional technology such as fabric knitting and circuit adapters. We also observed that there were topics overlapping among different graph clusters. We found that such overlapping sometimes shows the evolution of technology. For example, the topic about "packet, signal" appeared in both a smaller cluster and a larger cluster. The one in the smaller cluster discusses more foundational technology such as decoding and CRC checking; the larger cluster discussion is regarding more advanced/applied technology such as wireless communication. We also applied HierNMF2 to a set of litigated patents and identified several major domains of litigated patents, such as computer engineering, electrical engineering and telecommunications, etc.

We published the following papers based on our research and discovery on the summer challenge and hackathon data sets: [54, 55, 56, 57, 58].

4.6 Our XDATA Open Source Software: SmallK

SmallK (smallk.github.io) is a high performance software package for constrained low rank matrix approximation via the nonnegative matrix factorization (NMF). Algorithms for NMF compute the low rank factors of a matrix producing two nonnegative matrices whose product approximates the original matrix. The role of NMF in data analytics has been as significant as the singular value decomposition (SVD). However, due to nonnegativity constraints, NMF has far superior interpretability of its results for many practical problems such as image processing, chemometrics, bioinformatics, topic modeling for text analytics and many more. Our approach to solving the NMF nonconvex optimization problem has proven convergence properties and is one of the most efficient methods developed to date. The latest upgrades to SmallK include Docker and Vagrant virtual machine installations and other enhancements based on new operating system requirements

such as compatibility with Apple's OSX System Integrity Protection (SIP) in the Sierra version. Also, the website has a new face using the Sphinx documentation system.

Recently open sourced: MPI-FAUN! Both MPI and OPENMP implementations for MU, HALS and ANLS/BPP based NMF algorithms are now available. The implementations can run off the shelf or can be easily integrated into other source code. These are very highly tuned NMF algorithms to work on super computers. We have tested this software in NERSC (National Energy Research Scientific Computing Center) as well as OLCF (Oak Ridge Leadership Computing Facility) cluster. The openmp implementation is tested on many different linux variants with intel processors. The library works well for both sparse and dense matrices.

Recent Stats: since 2018/03/15: 55 views and 10 clones from smallk.gitub.io; 0 issues

Users of SmallK:
AFRL (Air Force Research Laboratory) - Rome Lab, Uncharted Inc., Toronto, Canada - Tile-based visual system with geolocated topic modeling for event analysis, USAF - Cyber Security Applications, GTRI (Georigia Tech Research Institute) - Cybersecurity, Information Protection, and Hardware Evaluation Research (CIPHER) Laboratory, CIA (Central Intelligence Agency) - Threat Analysis, USMC (United States Marine Corps) - HQMC-C4 (Headquarters Marine Corps - Command, Control, Communications and Computers), ORNL (Oak Ridge National Laboratory). Also a project funded by Clarkson Aerospace for ROTC training in next generation methods for EW: Cyber Spectrum Research and Technology Development Virtual Environment (CSpec-DVE), GTRI Dayton Field Office in collaboration with GTRI Atlanta, Information and Communications Laboratory (ICL), Software Engineering and (INNC). The purpose was to train coming generations of officers in modern data analytics techniques, and a Georgia Tech developed topic modeling software was delivered in mixed language architecture Matlab, Python, and Java. The software is based on the SmallK low rank approximation library available through the DARPA Open Catalog. The modular DVE Matlab software framework with GTRI ICL/SEAD developed Topic Modeling module, and other modules.

5 Conclusions

In this project, we studied NMF variants for three important tasks for big data analysis: NMF and DC-NMF for text clustering/topic modeling, SymNMF and HierSymNMF2 for graph clustering/community detection and JointNMF for hybrid clustering. For each task, we studied existing literatures extensively, proposed NMF formulations, designed efficient algorithms and conducted extensive experiments. We also constructed some new data sets. We have seen that NMF based methods usually had better clustering quality and comparable computational cost with other state-of-the-art algorithms. Besides clustering, NMF based methods provided valuable insights of the data. The simple and unified framework of NMF based method has allowed an unified, convergent and efficient solution framework (BCD) and made it flexible to combine multiple sources of information. The good interpretability of NMF has made it possible to apply NMF based clustering algorithms to problems such as text based link prediction and leader and activity detection.

NMF has been applied in many real world projects. Using the methods described in this project, we have generated results for many real world data analysis tasks such as summarizing people's

opinion on sustainable technologies via text mining, detecting ceasefire violations in Yemen via analyzing Telegram messages and identifying emerging, fading and evolution of technology via analysis of patents. The area of NMF is still developing. Some recent and ongoing work includes NMF based co-clustering and NMF based outlier detection.

References

[1] W. Xu, X. Liu, and Y. Gong, "Document Clustering Based on Non-negative Matrix Factorization," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, ser. SIGIR '03, New York, NY, USA: ACM, 2003, pp. 267–273.

[2] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.

[3] M. Hofree, J. P. Shen, H. Carter, A. Gross, and T. Ideker, "Network-based stratification of tumor mutations," *Nature Methods*, vol. 10, no. 11, pp. 1108–1115, Nov. 2013.

[4] A. Ozerov and C. Févotte, "Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 3, pp. 550–563, 2010.

[5] B. Drake, S. Lee-Urban, and H. Park, *Smallk is a C++/Python high-performance software library for nonnegative matrix factorization (nmf) and hierarchical and flat clustering using the nmf; current version 1.6.2*, `http://smallk.github.io/`, 2017.

[6] R. Kannan, G. Ballard, and H. Park, "A high-performance parallel algorithm for nonnegative matrix factorization," in *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '16, Barcelona, Spain: ACM, 2016, 9:1–9:11.

[7] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, ser. NIPS'00, Denver, CO: MIT Press, 2000, pp. 535–541.

[8] 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.

[9] J. Kim and H. Park, "Sparse nonnegative matrix factorization for clustering," Georgia Institute of Technology, Tech. Rep., 2008.

[10] D. Kuang and H. Park, "Fast Rank-2 Nonnegative Matrix Factorization for Hierarchical Document Clustering," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13, New York, NY, USA: ACM, 2013, pp. 739–747.

[11] D. Kuang, S. Yun, and H. Park, "SymNMF: Nonnegative low-rank approximation of a similarity matrix for graph clustering," *Journal of Global Optimization*, vol. 62, no. 3, pp. 545–574, Nov. 2014.

[12] N. Gillis, D. Kuang, and H. Park, "Hierarchical Clustering of Hyperspectral Images Using Rank-Two Nonnegative Matrix Factorization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 4, pp. 2066–2078, Apr. 2015.

[13] L. Li, G. Lebanon, and H. Park, "Fast bregman divergence nmf using taylor expansion and coordinate descent," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '12, Beijing, China: ACM, 2012, pp. 307–315.

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

[15] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, Award winning papers from the 19th International Conference on Pattern Recognition (ICPR)19th International Conference in Pattern Recognition (ICPR), vol. 31, no. 8, 651–666, 2010.

[16] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, 395–416, 2007.

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th. The Johns Hopkins University Press, 2013.

[18] 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.

[19] J. Kim and H. Park, "Fast Nonnegative Matrix Factorization: An Active-Set-Like Method and Comparisons," *SIAM J. Sci. Comput.*, vol. 33, no. 6, pp. 3261–3281, Nov. 2011.

[20] D. Kuang, C. Ding, and H. Park, "Symmetric Nonnegative Matrix Factorization for Graph Clustering," in *Proceedings of the 2012 SIAM International Conference on Data Mining*, ser. Proceedings, Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, Apr. 2012, pp. 106–117.

[21] N.-D. Ho, "Non-negative matrix factorization. algorithms and applications," PhD thesis, Université catholique de Louvain, 2008.

[22] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, Nov. 2002.

[23] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.

[24] Y. Xu, W. Yin, Z. Wen, and Y. Zhang, "An alternating direction algorithm for matrix completion with nonnegative factors," *Frontiers of Mathematics in China*, vol. 7, no. 2, pp. 365–384, 2012.

[25] 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.

[26] 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.

[27] 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.

[28] C.-J. Hsieh and I. S. Dhillon, "Fast coordinate descent methods with variable selection for non-negative matrix factorization," in *17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '11)*, 2011, pp. 1064–1072.

[29] 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.

[30] 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.

[31] C.-J. Lin, "On the convergence of multiplicative update algorithms for nonnegative matrix factorization," *IEEE Trans. on Neural Networks*, vol. 18, no. 6, pp. 1589–1596, 2007.

[32] F. G. Nicolas Gillis, "Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization," *Neural Computation*, vol. 24, no. 4, pp. 1085–1105, 2012.

[33] 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*, vol. 30, no. 2, pp. 713–730, 2008.

[34] 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.

[35] 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.

[36] 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.

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

[38] N. Gillis, "The why and how of nonnegative matrix factorization," in *Regularization, Optimization, Kernels, and Support Vector Machines*, J. Suykens, M. Signoretto, and A. Argyriou, Eds., Chapman & Hall/CRC, 2014, ch. 12, pp. 257–291.

[39] J. J. Whang, D. F. Gleich, and I. S. Dhillon, "Overlapping Community Detection Using Neighborhood-Inflated Seed Expansion," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 5, pp. 1272–1284, May 2016.

[40] A. Prat-Pérez, D. Dominguez-Sal, and J.-L. Larriba-Pey, "High Quality, Scalable and Parallel Community Detection for Large Real Graphs," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14, New York, NY, USA: ACM, 2014, pp. 225–236.

[41] J. Yang and J. Leskovec, "Overlapping Community Detection at Scale: A Nonnegative Matrix Factorization Approach," in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, ser. WSDM '13, New York, NY, USA: ACM, 2013, pp. 587–596.

[42] I. Dhillon, Y. Guan, and B. Kulis, "Weighted Graph Cuts without Eigenvectors A Multilevel Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 11, pp. 1944–1957, Nov. 2007.

[43] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford large network dataset collection*, `http://snap.stanford.edu/data`, Jun. 2014.

[44] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, Oct. 2013.

[45] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and Analysis of Online Social Networks," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '07, New York, NY, USA: ACM, 2007, pp. 29–42.

[46] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, "Group Formation in Large Social Networks: Membership, Growth, and Evolution," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '06, New York, NY, USA: ACM, 2006, pp. 44–54.

[47] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The Dynamics of Viral Marketing," *ACM Trans. Web*, vol. 1, no. 1, May 2007.

[48] dblp, *What is dblp?* `http://dblp.uni-trier.de/faq/What+is+dblp.html`, Accessed: 2015-06-29, 2015.

[49] ——, *Dblp: Computer science bibliography*, `http://dblp.uni-trier.de/`, Accessed: 2015-06-17, 2015.

[50] ——, *What do i find in dblp.xml?* `http://dblp.uni-trier.de/faq/What+do+I+find+in+dblp+xml.html`, Accessed: 2015-06-30, 2015.

[51] X. Wang, L. Tang, H. Gao, and H. Liu, "Discovering Overlapping Groups in Social Media," in *2010 IEEE International Conference on Data Mining*, Dec. 2010, pp. 569–578.

[52] X. Wang, L. Tang, H. Liu, and L. Wang, "Learning with multi-resolution overlapping communities," *Knowledge and Information Systems*, vol. 36, no. 2, pp. 517–535, Aug. 2013.

[53] T. Yang, R. Jin, Y. Chi, and S. Zhu, "Combining Link and Content for Community Detection: A Discriminative Approach," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09, New York, NY, USA: ACM, 2009, pp. 927–936.

[54] J. Choo, D. Lee, B. Dilkina, H. Zha, and H. Park, "To gather together for a better world: Understanding and leveraging communities in micro-lending recommendation," in *Proceedings of the 23rd international conference on World wide web*, ACM, 2014, pp. 249–260.

[55] J. Choo, C. Lee, D. Lee, H. Zha, and H. Park, "Understanding and promoting micro-finance activities in kiva. org," in *Proceedings of the 7th ACM international conference on Web search and data mining*, ACM, 2014, pp. 583–592.

[56] J. C.S.L.C.B.P.H.N.K.R.K.B.D.J. C. S. Shin M. Choi and H. Park, "STExNMF: Spatio-temporally exclusive topic discovery for anomalous event detection," in *ICDM '17: Proc. of the 17th IEEE Int. Conf. on Data Mining*, To appear, 2017.

[57] J. Choo, Y. Han, M. Hu, H. Kim, J. Nugent, F. Poggi, J. Stasko, and H. Park, "High-recall document retrieval from large-scale noisy documents via visual analytics based on targeted topic modeling," in *VAST '17: 2017 IEEE Conf. on Visual Analytics Science and Technology*, To appear, 2017.

[58] B. Drake, T. Huang, A. Beavers, R. Du, and H. Park, "Event detection based on nonnegative matrix factorization: Ceasefire violation, environmental, and malware events," in *Proceedings of the 8th International Conference on Applied Human Factors and Ergonomics (AHFE 2017), Human Factors in Cybersecurity, Springer, invited paper*, to appear, 2017.