Data Weaving: Scaling Up the State-Of-The-Art in Data Clustering

Ron Bekkerman and Martin Scholz
HP Laboratories
HPL-2008-38R2

Keyword(s):
Information-theoretic clustering, multi-modal clustering, parallel and distributed data mining

Abstract:
The enormous amount and dimensionality of data processed by modern data mining tools require effective, scalable unsupervised learning techniques. Unfortunately, the majority of previously proposed clustering algorithms are either effective or scalable. This paper is concerned with information-theoretic clustering (ITC) that has historically been considered the state-of-the-art in clustering multi-dimensional data. Most existing ITC methods are computationally expensive and not easily scalable. Those few ITC methods that scale well (using, e.g., parallelization) are often out-performed by the others, of an inherently sequential nature. First, we justify this observation theoretically. We then propose data weaving--a novel method for parallelizing sequential clustering algorithms. Data weaving is intrinsically multi-modal--it allows simultaneous clustering of a few types of data (modalities). Finally, we use data weaving to parallelize multi-modal ITC, which results in proposing a powerful DataLoom algorithm. In our experimentation with small datasets, DataLoom shows practically identical performance compared to expensive sequential alternatives. On large datasets, however, DataLoom demonstrates significant gains over other parallel clustering methods. To illustrate the scalability, we simultaneously clustered rows and columns of a contingency table with over 120 billion entries.

External Posting Date: April 6, 2009 [Fulltext] Approved for External Publication

Internal Posting Date: August 21, 2008 [Fulltext]

Published in ACM CIKM, Conference on Information & Knowledge Management, Napa, CA Oct 27, 2008

Data Weaving: Scaling Up the State-Of-The-Art in Data Clustering

Ron Bekkerman
HP Laboratories
Palo Alto, CA 94304, USA
ron.bekkerman@hp.com

Martin Scholz
HP Laboratories
Palo Alto, CA 94304, USA
scholz@hp.com

ABSTRACT

The enormous amount and dimensionality of data processed by modern data mining tools require effective, scalable unsupervised learning techniques. Unfortunately, the majority of previously proposed clustering algorithms are *either* effective *or* scalable. This paper is concerned with information-theoretic clustering (ITC) that has historically been considered the state-of-the-art in clustering multi-dimensional data. Most existing ITC methods are computationally expensive and not easily scalable. Those few ITC methods that scale well (using, e.g., parallelization) are often outperformed by the others, of an inherently sequential nature. First, we justify this observation theoretically. We then propose *data weaving*—a novel method for parallelizing sequential clustering algorithms. Data weaving is intrinsically *multi-modal*—it allows simultaneous clustering of a few types of data (*modalities*). Finally, we use data weaving to parallelize multi-modal ITC, which results in proposing a powerful *DataLoom* algorithm. In our experimentation with small datasets, DataLoom shows practically identical performance compared to expensive sequential alternatives. On large datasets, however, DataLoom demonstrates significant gains over other parallel clustering methods. To illustrate the scalability, we simultaneously clustered rows and columns of a contingency table with over 120 billion entries.

Categories and Subject Descriptors

I.5 [**Pattern Recognition**]: Clustering; D.1 [**Programming Techniques**]: Concurrent Programming—*Parallel programming*

General Terms

Algorithms

Keywords

Information-theoretic clustering, multi-modal clustering, parallel and distributed data mining

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.

1. INTRODUCTION

It becomes more and more apparent that processing gigabytes and terabytes of information is a part of our everyday routine. While data mining technology is already mature enough for tasks of that magnitude, we—data mining practitioners—are not always prepared to use the technology in full. For example, when a practitioner faces a problem of clustering a hundred million data points, a typical approach is to apply the simplest method possible, because it is hard to believe that fancier methods can be feasible. Whoever adopts this approach makes two mistakes:

• **Simple clustering methods are not always feasible.** Let us consider, for example, a simple online clustering algorithm (which, we believe, is machine learning folklore): first initialize k clusters with one data point each, then iteratively assign the rest of points into their closest clusters (in the Euclidean space). Even for small values of k (say, $k = 1000$), such an algorithm may work for hours on a modern PC. The results would however be quite unsatisfactory, especially if our data points are 100,000-dimensional vectors.

• **State-of-the-art clustering methods *can* scale well**, which we aim to show in this paper.

With the deployment of large computational facilities (such as Amazon.com's EC2, IBM's BlueGene, HP's XC), the parallel computing paradigm is probably the only currently available option for addressing gigantic data processing tasks. Parallel methods become an integral part of any data processing system, and thus gain special importance (e.g., some universities are currently introducing parallel methods to their core curricula [19]).

Despite that data clustering has been in the focus of the parallel and distributed data mining community for more than a decade, not many clustering algorithms have been parallelized, and not many software tools for parallel clustering have been built (see Section 3.1 for a short survey). Apparently, most of the parallelized clustering algorithms are fairly simple. There are two families of data clustering methods that are widely considered as very powerful:

• **Multi-modal (or multivariate) clustering** is a framework for simultaneously clustering a few types (or *modalities*) of the data. Example: construct a clustering of Web pages, together with a clustering of words from those pages, as well as a clustering of URLs hyperlinked from those pages. It is commonly believed that multi-modal clustering is able to achieve better results than traditional, uni-modal methods. The two-modal case (usually called *co-clustering* or *double clustering*) has been widely explored in the literature (see [14, 12]), however, a more general m-modal case

has only recently attracted close attention of the research community (see [17, 1]), probably because of its computational cost.

• **Information-theoretic clustering (ITC)** (see, e.g. [30]) is an adequate solution to clustering highly multi-dimensional data, such as documents or genes. ITC methods perform global optimization of an information-theoretic objective function. For the details, see Section 2.

Many global optimization methods are greedy—those methods are *sequential* in their essence, and therefore are difficult to parallelize. In contrast, local optimization methods are often easily parallelizable. Many popular clustering algorithms, such as k-means, belong to the latter category. Unfortunately, most of them are not very effective on large multi-dimensional datasets. In the text domain, for example, k-means usually ends up with one huge cluster and a few tiny ones.[1]

One approach to solving a global optimization problem is to break it down into a set of local optimizations. Dhillon et al. [12] applied this approach to perform an *information-theoretic co-clustering (IT-CC)*. In Section 3.2, we show a fairly straightforward way of parallelizing their algorithm. The IT-CC algorithm turns out to be very conservative in optimizing the (global) clustering objective, such that it gets often stuck in local optima. In Section 3.3, we discuss a *sequential co-clustering (SCC)* method, and show analytically that it is more aggressive in optimizing the objective.

In Section 4, we propose a new scheme for parallelizing sequential clustering methods, called *data weaving*. This mechanism works as a loom: it propagates the data through a rack of machines, gradually weaving a "fabric" of clusters. We apply this mechanism to parallelizing the SCC method, which leads to constructing a highly scalable, information-theoretic, multi-modal clustering algorithm, called *DataLoom*.

In the experimentation part of our paper (Section 5) we first compare DataLoom with its original, non-parallel version (SCC), as well as with IT-CC and two more baseline methods on four small datasets (including the benchmark 20 Newsgroups). We show that the parallelization does not compromise the clustering performance. Finally, we apply DataLoom to two large datasets: RCV1 [22], where we cluster documents and words, and Netflix KDD'07 Cup data,[2] where we cluster customers and movies. If represented as contingency tables, both datasets contain billions of entries. On both of them, DataLoom significantly outperforms the parallel IT-CC algorithm. To our knowledge, co-clustering experiments of that scale have not been reported previously.

2. INFORMATION-THEORETIC CLUSTERING

Over the past decade, information-theoretic clustering methods have proven themselves to be the state-of-the-art in clustering highly multi-dimensional data. In this paper, we focus on *hard* clustering (a many-to-one mapping of data points to cluster identities), as opposed to *soft* clustering (a many-to-many mapping, where each data point is assigned a probability distribution over cluster identities). Hard clustering can be viewed as a lossy compression scheme—this observation opens a path to applying various information-theoretic methods to clustering. Examples include the application of the minimum description length principle [6] and rate-distortion theory [9].

The latter led to proposing the powerful Information Bottleneck (IB) principle by Tishby et al. [34], and then to dozens of its extensions. In Information Bottleneck, a random variable X is clustered with respect to an interacting variable Y: the clustering $\tilde{X}$ is represented as a low-bandwidth channel (a *bottleneck*) between the input signal X and the output signal Y. This channel is constructed to minimize the communication error while maximizing the compression:

$$\max \left[I(\tilde{X};Y) - \beta I(\tilde{X};X) \right], \tag{1}$$

where I is a Mutual Information (MI), and β is a Lagrange multiplier. A variety of optimization procedures have been derived for the Information Bottleneck principle, including agglomerative [31], divisive [2], sequential (flat) [30] methods, and a hybrid of them [1].

Friedman et al. [16] generalize the IB principle to a multivariate case. In its simplest form, for clustering two variables X and Y, the generalization is relatively straightforward: a channel $X \leftrightarrow \tilde{X} \leftrightarrow \tilde{Y} \leftrightarrow Y$ is constructed to optimize the objective

$$\max \left[I(\tilde{X};\tilde{Y}) - \beta_1 I(\tilde{X};X) - \beta_2 I(\tilde{Y};Y) \right]. \tag{2}$$

When more than two variables are clustered, the mutual information $I(\tilde{X};\tilde{Y})$ is generalized into its multivariate version, called multi-information. The complexity of computing multi-information grows exponentially while adding more variables, and is therefore restrictive in practical cases even for only three variables.

Information-theoretic co-clustering (IT-CC) was proposed by Dhillon et al. [12] as an alternative to multivariate IB, for the two-variate case when the numbers of clusters $|\tilde{X}|$ and $|\tilde{Y}|$ are fixed. In this case, it is natural to drop the compression constraints $I(\tilde{X};X)$ and $I(\tilde{Y};Y)$ in Equation (2), and directly minimize the information loss:

$$\min \left[I(X;Y) - I(\tilde{X};\tilde{Y}) \right] = \max I(\tilde{X};\tilde{Y}), \tag{3}$$

when $I(X;Y)$ is a constant for a given dataset. To optimize this objective, Dhillon et al. proposed an elegant optimization method that resembles the traditional k-means, while the latter has a less powerful l^2 objective.

Bekkerman et al. [1] generalize IT-CC to the multivariate case, while avoiding the trap of multi-information: they approximate it with a (weighted) sum of pairwise MI terms:

$$\max \left[\sum_{e_{ij} \in \mathcal{E}} w_{ij} I(\tilde{X}_i;\tilde{X}_j) \right],$$

where the data variables $\{X_1, \ldots, X_m\}$ are organized in an *interaction graph* $G = (\mathcal{X}, \mathcal{E})$,[3] with edges e_{ij} corresponding

[1] The traditional k-means assigns instances to clusters based on the Euclidean distances between points and centroids. Since text is usually sparse and high-dimensional, documents typically have only few terms in common. As a consequence, the l^2 norms of terms and centroids often dominate in the calculation of their Euclidean distances. Since the l^2 norms of centroids naturally decrease with increasing the cluster size, instances tend to be re-assigned to clusters that are already large, and smaller clusters disappear over time.

[2] `http://cs.uic.edu/~liub/Netflix-KDD-Cup-2007.html`

[3] Lauritzen [21] presents the interaction graph as a generalization of a graphical model.

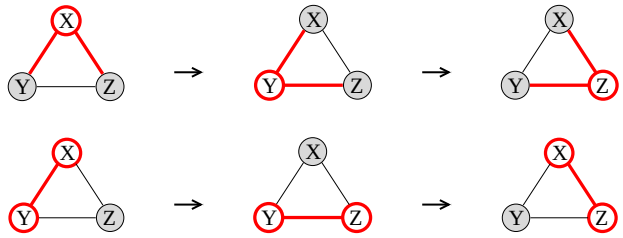

Figure 1: Difference between ICM (upper) and CWO (lower) optimization methods (nodes that are being optimized are unshaded bold). ICM iterates over G's nodes (in round-robin) and optimize each of them based on its Markov blanket. CWO iterates over cliques in G (edges, in the simplest case) and locally optimizes the corresponding model while ignoring the rest of the interaction graph.

to pairs of interacting variables (X_i, X_j). Weights w_{ij} are chosen to bring MI terms to the same scale. Bekkerman et al. propose a complex optimization method that utilizes the Iterative Conditional Modes (ICM) [4] for traversing the graph G, and then performs a hybrid hierarchical/sequential clustering step for each of G's nodes. An illustration of ICM is given in Figure 1 (upper).

We notice that the extra parametrization (through weights w_{ij}) can be avoided by changing the graph traversal scheme: instead of iterating over nodes (as in ICM), we can iterate over edges e_{ij} and maximize only one MI term $I(\tilde{X}_i; \tilde{X}_j)$ at a time (see the lower part of Figure 1). We call our method *Clique-Wise Optimization (CWO)*—it is analogous to *pairwise training* [33] in a supervised learning setup.

3. PARALLEL CLUSTERING

Parallel and distributed data mining is a very active field of research [23] that covers a variety of disciplines, so we want to narrow down the scope of this section. Although our methods are vaguely related to distributed clustering, we would like to point out that this paper does not address problems such as mining geographically distributed data, privacy preservation due to using insecure networks, or aspects of fault recovery. Instead, we are "just" concerned with scaling up state-of-the-art clustering algorithms to the very large amounts of data we deal with, while we assume a "shared nothing" cluster of computers connected via a high bandwidth local area network. Note that the clustering problem can be large-scale along several dimensions. Often not only the number of data instances is very large, but data is also of very high dimensionality; for example, ten thousands of features is common even for small text corpora. Whenever the task of clustering data collections requires to capture the underlying structure of a dataset at a fine level, using a very large number of clusters is also common. Our goal is to reduce the total computational costs to a tractable level in order to get the best possible clustering results on very large data collections.

3.1 Goals and Related Work

Among the early approaches explicitly mentioning and addressing the scalability problem along all three of these dimensions is Canopy [26], a non-distributed clustering algorithm that avoids many expensive distance computations by aggregating objects at a coarse level; only objects in a common "Canopy" are assumed to be close enough to potentially be in the same cluster. Scaling up drastically may compromise the data mining results, however, (and unlike ITC) the method requires an underlying distance metric.

Several authors addressed the scalability issue of clustering by parallelizing specific algorithms, most prominently k-means [20, 13] including its generalizations [15] that cover e.g., the EM algorithm. The parallelization strategies exploit the stage-wise nature and mathematical properties that allow to compute global from local solutions: Each node is responsible for a subset of the data. It computes the closest cluster for each of its instances, computes the new local cluster means (or parameters for EM, respectively), communicates these means to a master who aggregates them, and distributes the aggregated centroids (or parameters) for the next iteration. This parallelization procedure yields algorithms that compute identical results as their non-parallel counterparts.

On the technical side, the literature usually shows that these algorithms can be realized on top of a specific low-level communication framework [15, 13, 18] running on a "shared nothing" cluster, but they clearly are not limited to this kind of architecture. It has recently been discussed that the same kind of parallelization works very well in combination with the popular MapReduce paradigm [10]. Parallelizing the clustering algorithms k-means and EM (mixture of Gaussians) via MapReduce is covered in [8].

Parallelization of clustering algorithms is still an active field of research. As discussed above, in some fields, like density-based clustering, a good understanding of how to parallelize has already been gained, so the research focuses on rather specific cases like spatial aspects and structured data, with a clear bias towards distributed (as opposed to parallel) data mining [35, 7]. For information-theoretic clustering this research is still in its infancy. A clustering algorithm for specific distributed data problems that employs information-theoretic objectives has recently been proposed by [11], but it does not discuss efficient *parallelization* of ITC. To the best of our knowledge, a systematic study of the practically highly relevant parallelization of information-theoretic clustering is still lacking. As we will see in Section 3.2, adopting the strategies sketched above already allows for a straightforward parallelization of an important ITC algorithm.

3.2 Parallel IT-CC

Dhillon et al.'s information-theoretic co-clustering (IT-CC) [12] is a k-means-style algorithm that *locally* optimizes the global information-theoretic objective function (3). We first briefly sketch the formal background of IT-CC, before proposing its parallelization.

The goal of IT-CC is to approximate a given joint probability distribution p over two modalities X and Y with a "simpler" distribution q, where the statistical dependencies are captured in a lower-dimensional cluster space:

$$q(x, y) := p(x) \cdot p(y) \cdot \frac{p(\tilde{x}, \tilde{y})}{p(\tilde{x}) \cdot p(\tilde{y})}, \tag{4}$$

where $x \in X$, $y \in Y$, $p(x)$ and $p(y)$ denote marginals; $\tilde{x}$ and $\tilde{y}$ are the corresponding clusters of x and y, respectively; and $p(\tilde{x})$ and $p(\tilde{y})$ are marginals of these clusters. Dhillon et al. show that optimization of the objective func-

tion (3) is equivalent to the minimization of the KL divergence $D_{KL}(p(X,Y)||q(X,Y))$ between the joint distribution p and its approximation q.

Like many other co-clustering algorithms, IT-CC alternates iterations that update the clustering assignments $\tilde{X}$ and $\tilde{Y}$. The reason we think of IT-CC as a k-means-style algorithm is that it first assigns all data points to their closest clusters, and then it recomputes cluster representatives based on the data points now contained in each cluster.

Let us focus on a IT-CC iteration where the clustering $\tilde{X}$ is updated given the clustering $\tilde{Y}$ (the opposite case is symmetric.) Unlike the traditional k-means that uses the Euclidian distance metric, IT-CC defines the proximity of a data point x to a cluster $\tilde{x}$ in terms of the KL divergence between $p(Y|x)$ and $q(Y|\tilde{x})$, where the latter is computed using

$$q(y|\tilde{x}) = q(y|\tilde{y}) \cdot q(\tilde{y}|\tilde{x}). \tag{5}$$

During the data point assignment process, the conditionals $q(Y|\tilde{x})$ do not change, thus playing the role of the centroids.

Dhillon et al. prove that the co-clustering strategy of assigning data points x to clusters $\tilde{x}$ by minimizing the local objective $D_{KL}(p(Y|x)||q(Y|\tilde{x}))$, monotonically decreases the global objective function, which guarantees the algorithm's convergence. The following transformations illustrate how to simplify computations without changing the optimization problem, that is, without changing the total order of its solutions. We remove terms that are constants in the context of optimizing cluster assignment $\tilde{X}$, and rewrite

$$\begin{aligned}
& \arg\min_{\tilde{x}} D_{KL}(p(Y|x)||q(Y|\tilde{x})) \\
= \; & \arg\min_{\tilde{x}} \sum_{y} p(y|x) \log \frac{p(y|x)}{q(y|\tilde{x})} \\
& \textit{(substituting Equation (5) for } q(y|\tilde{x})\textit{:)} \\
= \; & \arg\max_{\tilde{x}} \sum_{y} p(y|x) \log(q(y|\tilde{y})q(\tilde{y}|\tilde{x})) \\
= \; & \arg\max_{\tilde{x}} \sum_{\tilde{y}} p(\tilde{y}|x) \log q(\tilde{y}|\tilde{x}) \\
= \; & \arg\max_{\tilde{x}} \sum_{\tilde{y}} p(\tilde{y}|x) \log p(\tilde{y}|\tilde{x}).
\end{aligned} \tag{6}$$

The above transformation shows that rather than computing the centroids $q(Y|\tilde{x})$, the algorithm only needs to compute $q(\tilde{x},\tilde{y}) = p(\tilde{x},\tilde{y})$ for each cluster pair $\tilde{x}$ and $\tilde{y}$ at each round.

We argue that the same simplification allows to select an optimal clustering $\tilde{X}_{opt}$ from a set of candidate clusterings $\tilde{\mathcal{X}} \subset 2^X$ (given $\tilde{Y}$) by *only* referring to cluster joints $p(\tilde{x},\tilde{y})$. Let $q_{(\tilde{X},\tilde{Y})}$ be the distribution q induced by a specific pair of clusterings $\tilde{X}$ and $\tilde{Y}$. We have:

$$\begin{aligned}
\tilde{X}_{opt} &= \arg\min_{\tilde{X}\in\tilde{\mathcal{X}}} D_{KL}(p(X,Y)||q_{(\tilde{X},\tilde{Y})}(X,Y)) \\
&= \arg\min_{\tilde{X}\in\tilde{\mathcal{X}}} \sum_{x\in X, y\in Y} p(x,y) \log \frac{p(x,y)}{q_{(\tilde{X},\tilde{Y})}(x,y)} \\
&= \arg\max_{\tilde{X}\in\tilde{\mathcal{X}}} \sum_{\tilde{x}\in\tilde{X}} p(\tilde{x}) \sum_{\tilde{y}\in\tilde{Y}} p(\tilde{y}|\tilde{x}) \log q(\tilde{y}|\tilde{x}) \\
&= \arg\max_{\tilde{X}\in\tilde{\mathcal{X}}} \sum_{\tilde{x}\in\tilde{X}} p(\tilde{x}) \sum_{\tilde{y}\in\tilde{Y}} p(\tilde{y}|\tilde{x}) \log p(\tilde{y}|\tilde{x})
\end{aligned} \tag{7}$$

as our new, equivalent formulation of the IT-CC optimization problem. The function in Equation (7) preserves even the correct total order of the candidate clusterings with respect to the mutual information (3).

Following the outline of parallel k-means (Section 3.1) and the description above, we can adapt IT-CC to the parallel case as follows: We alternate the optimization of $\tilde{X}$ and $\tilde{Y}$. During each of these optimizations the parallel processes hold disjoint subsets of the data. We will just describe the case of computing a new clustering of X; clustering Y works analogously. Process i will hold the data for elements $X^{(i)} \subset X$. All cluster "centroids" $p(\tilde{x},\tilde{y})$ are distributed to all nodes, where the new cluster assignments are computed based on the KL divergence (6). Given the new assignments, each process i computes the local joints $q^{(i)}(\tilde{x},\tilde{y}) = \sum_{x\in\tilde{x}\cap X^{(i)}} p(x,\tilde{y})$ for each (new) cluster $\tilde{x}$, and broadcasts them to a master node. The master computes the new global "centroids" $q(\tilde{x},\tilde{y}) = \sum_i q^{(i)}(\tilde{x},\tilde{y})$. They can then be broadcasted to the nodes again to start the next round of refining $\tilde{X}$, or the algorithm can switch to re-cluster Y instead. Note that this process yields exactly the same results as in the non-parallelized case.

We consider the parallel IT-CC algorithm as a strong baseline for the DataLoom algorithm proposed in a Section 4. Before moving on, let us discuss the potential of DataLoom by taking a closer look at the difference between IT-CC and the (non-parallelized) sequential information bottleneck.

3.3 Sequential Co-clustering

DataLoom originates from a multi-modal version of the *sequential Information Bottleneck (sIB)* algorithm [30]. In sIB, at its initialization step, all data points are uniformly at random assigned into clusters. Then, a random permutation of all the data points is constructed, each element of which is pulled out of its cluster and iteratively assigned into any other cluster. It is finally left in the cluster such that the objective function (1) is maximized. The algorithm is executed until its full convergence.

We consider the multi-modal variation of sIB (we call it *sequential co-clustering (SCC)*), which iterates over the data modalities organized in an interaction graph (see Section 2). At each iteration, it applies the sIB's optimization procedure to maximize the co-clustering objective (3). It improves clusterings by continuously updating cluster memberships of individual data points. To decide whether to change a cluster membership, it directly evaluates the objective.

PROPOSITION 3.1. *The set of clustering pairs $(\tilde{X},\tilde{Y})$ that are local optima of SCC are a subset of the clustering pairs that are local optima of IT-CC.*

PROOF. It is sufficient to show that, whenever IT-CC reads a pair of clusterings $(\tilde{X},\tilde{Y})$ and outputs a pair of clusterings $(\tilde{X}',\tilde{Y}')$ with a higher score of the objective function, SCC will improve the objective function on $(\tilde{X},\tilde{Y})$ as well. We will just discuss this for the case of re-clustering X; the case of re-clustering $\tilde{Y}$ can be shown analogously.

By design, the only case in which SCC fails to improve the objective is the case in which it does not change *any* cluster memberships. Let us show that this *cannot* happen for any input that is not a local optimum of IT-CC. Whenever the output $(\tilde{X}^*,\tilde{Y})$ of IT-CC improves the objective function over the input $(\tilde{X},\tilde{Y})$, we know that IT-CC has changed the cluster membership of at least one element x', say, from $\tilde{x}'$ to

$\tilde{x}^*$. In terms of the local objective function (Equation (6)), this implies that

$$\sum_{\tilde{y}\in\tilde{Y}} p(\tilde{y}|x') \log q(\tilde{y}|\tilde{x}') < \sum_{\tilde{y}\in\tilde{Y}} p(\tilde{y}|x') \log q(\tilde{y}|\tilde{x}^*). \quad (8)$$

We define the clustering $\tilde{X}^*$ identically to $\tilde{X}$, except for moving x' to $\tilde{x}^*$. Let $q := q_{(\tilde{X},\tilde{Y})}$ refer to the approximation of p induced by $(\tilde{X},\tilde{Y})$, and $q^* := q_{(\tilde{X}^*,\tilde{Y})}$ be the corresponding distribution based on the clustering in which x' was moved. We will show, using a similar technique as in [12], that SCC cannot be stuck in a local optimum at this point, because its objective function favors $(\tilde{X}^*,\tilde{Y})$ over $(\tilde{X},\tilde{Y})$, so SCC would at least also move x' from $\tilde{x}'$ to $\tilde{x}^*$. We will use the following notation that always returns the old cluster "centroids" based on $(\tilde{X},\tilde{Y})$, although it already considers x' part of $\tilde{x}^*$:

$$\hat{q}(\tilde{y}|x) := \begin{cases} q(\tilde{y}|\tilde{x}), & \text{if } x \neq x', \tilde{x} \text{ being the cluster of } x \\ q(\tilde{y}|\tilde{x}^*), & \text{if } x = x' \end{cases}$$

For $\tilde{x} \in \tilde{X}^*$, all $x \in \tilde{x}$ share the same value $\hat{q}(\tilde{y}|x)$, so we can refer to this common value by writing $\hat{q}(\tilde{y}|\tilde{x})$ in this case.

Let us start with the value of the SCC's global objective function (Equation (7)) for the clustering pair $(\tilde{X},\tilde{Y})$, and show that it can be increased by moving x' into $\tilde{x}^*$:

$$\begin{aligned}
&\sum_{\tilde{x}\in\tilde{X}} p(\tilde{x}) \sum_{\tilde{y}\in\tilde{Y}} p(\tilde{y}|\tilde{x}) \log q(\tilde{y}|\tilde{x}) \quad (9)\\
=\ &\left(\sum_{\tilde{x}\in\tilde{X}} \sum_{x\in\tilde{x}\setminus\{x'\}} p(x) \sum_{\tilde{y}\in\tilde{Y}} p(\tilde{y}|x) \log q(\tilde{y}|\tilde{x})\right)\\
&+p(x') \sum_{\tilde{y}\in\tilde{Y}} p(\tilde{y}|x') \log q(\tilde{y}|\tilde{x}')\\
<\ &\left(\sum_{\tilde{x}\in\tilde{X}} \sum_{x\in\tilde{x}\setminus\{x'\}} p(x) \sum_{\tilde{y}\in\tilde{Y}} p(\tilde{y}|x) \log q(\tilde{y}|\tilde{x})\right)\\
&+p(x') \sum_{\tilde{y}\in\tilde{Y}} p(\tilde{y}|x') \log q(\tilde{y}|\tilde{x}^*)\\
=\ &\sum_{\tilde{x}\in\tilde{X}} \sum_{x\in\tilde{x}} p(x) \sum_{\tilde{y}\in\tilde{Y}} p(\tilde{y}|x) \log \hat{q}(\tilde{y}|x)\\
=\ &\sum_{\tilde{x}\in\tilde{X}^*} \sum_{\tilde{y}\in\tilde{Y}} p(\tilde{x},\tilde{y}) \log \hat{q}(\tilde{y}|\tilde{x})\\
=\ &\sum_{\tilde{x}\in\tilde{X}^*} q^*(\tilde{x}) \sum_{\tilde{y}\in\tilde{Y}} q^*(\tilde{y}|\tilde{x}) \log \hat{q}(\tilde{y}|\tilde{x})\\
\leq\ &\sum_{\tilde{x}\in\tilde{X}^*} q^*(\tilde{x}) \sum_{\tilde{y}\in\tilde{Y}} q^*(\tilde{y}|\tilde{x}) \log q^*(\tilde{y}|\tilde{x})\\
=\ &\sum_{\tilde{x}\in\tilde{X}^*} p(\tilde{x}) \sum_{\tilde{y}\in\tilde{Y}} p(\tilde{y}|\tilde{x}) \log q^*(\tilde{y}|\tilde{x}) \quad (10)
\end{aligned}$$

The first inequality holds due to Equation (8), the following steps just rearrange the summation over all cluster pairs and exploit the identity of joint cluster distributions $p(\tilde{x},\tilde{y})$ and $q^*(\tilde{x},\tilde{y})$. The second inequality holds due to the non-negativity of the KL divergence. The last step just rearranges terms again.

Together this proves that SCC would reassign at least x', because the value of the objective function after moving the single example x' (Equation 10) is strictly higher than the original value for $\tilde{X}$ (Equation 9). □

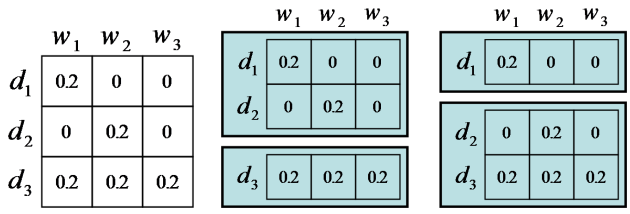

Figure 2: An illustration to the difference between IT-CC and SCC optimization procedures, used in Proposition 3.2.

PROPOSITION 3.2. *The subset relationship described in Proposition 3.1 is strict.*

PROOF. We prove this by presenting an example where IT-CC gets stuck in a local optimum which SCC is able to overcome. We look at three documents with the following sets of words: $d_1 = \{w_1\}$, $d_2 = \{w_2\}$, and $d_3 = \{w_1, w_2, w_3\}$. Initially, the first two documents are in cluster $\tilde{d}_1$, while the third one is in another cluster $\tilde{d}_2$. For simplicity, we assume that each word is in a separate cluster over the "word modality" W. Figure 2 shows the joint probability matrix p (left) and the initial aggregation to clusters (middle). The conditional distributions are hence $p(\tilde{W}|\tilde{d}_1) = (0.2, 0.2, 0)$ (upper cluster) and $p(\tilde{W}|\tilde{d}_2) = (0.2, 0.2, 0.2)$ (lower cluster). It can easily be verified by applying Equation (6) that IT-CC will not move any document. However, SCC will move either d_1 or d_2 into the second cluster. By applying this modification, SCC will almost double the mutual information (3) from about 0.17 (middle) to about 0.32 (right). □

Propositions 3.1 and 3.2 reveal that IT-CC gets stuck in local minima more often than SCC. Moreover, from the proofs we can conclude that generally, at the level of updating individual cluster memberships, IT-CC is more conservative. More specifically, this result suggests that the sequential strategy might both converge faster (because in every iteration it will do a number of updates that IT-CC misses) and to a better local optimum. We leave the verification of this conjecture to the empirical part of this paper.

4. THE DATALOOM ALGORITHM

Parallelization of sequential co-clustering (SCC) is allowed based on the following fairly straightforward consideration: mutual information $I(\tilde{X};\tilde{Y})$, which is the objective function of SCC, has the additive property over either of its arguments. That is, when SCC optimizes $\tilde{X}$ with respect to $\tilde{Y}$, and a data point $x' \in \tilde{x}'$ asks to move to cluster $\tilde{x}^*$, only the portion of the mutual information that corresponds to clusters $\tilde{x}'$ and $\tilde{x}^*$ is affected. Indeed, by definition,

$$\begin{aligned}
&I(\tilde{X};\tilde{Y}) = \sum_{\tilde{x}} \sum_{\tilde{y}} p(\tilde{x},\tilde{y}) \log \frac{p(\tilde{x},\tilde{y})}{p(\tilde{x})p(\tilde{y})}\\
&= \sum_{\tilde{x}\neq\tilde{x}'\wedge\tilde{x}\neq\tilde{x}^*} \sum_{\tilde{y}} p(\tilde{x},\tilde{y}) \log \frac{p(\tilde{x},\tilde{y})}{p(\tilde{x})p(\tilde{y})} +\\
&\sum_{\tilde{y}} \left[p(\tilde{x}',\tilde{y}) \log \frac{p(\tilde{x}',\tilde{y})}{p(\tilde{x}')p(\tilde{y})} + p(\tilde{x}^*,\tilde{y}) \log \frac{p(\tilde{x}^*,\tilde{y})}{p(\tilde{x}^*)p(\tilde{y})}\right].
\end{aligned}$$

To check whether or not moving x' into $\tilde{x}^*$ increases our objective function, it is sufficient to compute the delta between

Input:
G – interaction graph of nodes $\{X_1, \ldots, X_m\}$ and edges $\mathcal{E}$
$p(X_i, X_j)$ – pairwise joint distributions, for each edge e_{ij}
l – number of optimization iterations
Output:
Clusterings $\{\tilde{X}_1, \ldots, \tilde{X}_m\}$

Initialization:
For each node X **do**
  Assign values x to clusters $\tilde{x}$ uniformly at random
Main loop:
For each iteration $(1, \ldots, l)$ **do**
  For each edge $e_{ij} = (X_i, X_j)$ **do**
    For each ordering $(X, Y) \in \{(X_i, X_j), (X_j, X_i)\}$ **do**
      For each random restart **do**
        Compose pairs of clusters $(\tilde{x}, \tilde{x}')$ uniformly at random
        Assign each pair $(\tilde{x}, \tilde{x}')$ to a slave process
        Build input $\{p(x, \tilde{Y}) | x \in (\tilde{x}, \tilde{x}')\}$ for each slave
        Repeat
          Run slave processes
          Wait and monitor
          If system failure **then** kill all slave processes
        Until all slave processes successfully completed
        Compute $I(\tilde{X}; \tilde{Y})$
      Choose clustering $\tilde{X}$ with maximal $I(\tilde{X}; \tilde{Y})$ among all random restarts

Algorithm 1: Master process.

its value before the move and after the move. Again, only terms that correspond to $\tilde{x}'$ and $\tilde{x}^*$ are involved in the delta computation. Also, the marginals $p(\tilde{y})$ cancel out:

$$\begin{aligned}\Delta I(\tilde{X}; \tilde{Y}) &= I_{\text{after}}(\tilde{X}; \tilde{Y}) - I_{\text{before}}(\tilde{X}; \tilde{Y}) \\ &= \sum_{\tilde{y}} \Bigg[p\left(\tilde{x}' \setminus \{x'\}, \tilde{y}\right) \log \frac{p\left(\tilde{x}' \setminus \{x'\}, \tilde{y}\right)}{p\left(\tilde{x}' \setminus \{x'\}\right)} + \\ & p\left(\tilde{x}^* \cup \{x'\}, \tilde{y}\right) \log \frac{p\left(\tilde{x}^* \cup \{x'\}, \tilde{y}\right)}{p\left(\tilde{x}^* \cup \{x'\}\right)} - \\ & p(\tilde{x}', \tilde{y}) \log \frac{p(\tilde{x}', \tilde{y})}{p(\tilde{x}')} - p(\tilde{x}^*, \tilde{y}) \log \frac{p(\tilde{x}^*, \tilde{y})}{p(\tilde{x}^*)} \Bigg]. \quad (11)\end{aligned}$$

This brings us to the idea that probing the moves $x' \to \tilde{x}^*$ can be performed in parallel if all the clusters of $\tilde{X}$ are split into disjoint pairs. Each probing like that can be then executed using a separate process, after which the processes can exchange their data. Since the communication is generally expensive, it is beneficial to test *all* elements of both $\tilde{x}'$ and $\tilde{x}^*$. If the probe shows that the objective can be increased, the element is immediately moved from its cluster into another. Using this approach, we lose one ingredient of SCC: data points do not necessarily move into the cluster such that the objective function is *maximized*, but only *increased*. Despite that, intuitively, such a loss might look crucial, Bekkerman et al. [3] empirically show that both approaches are comparable, as soon as the number of optimization steps is about the same. The latter can be achieved by iterating over *all* the cluster pairs.

The DataLoom algorithm consists of a master process and $\frac{k}{2}$ slave processes (where k is the number of clusters). The master's algorithm is shown in Algorithm 1, the slave's in Algorithm 2. After constructing the initial set of cluster

Input:
$(\tilde{x}, \tilde{x}')$ – two clusters from $\tilde{X}$
$p(x, \tilde{Y})$ – rows of probability table $p(X, \tilde{Y})$ for $\forall x \in (\tilde{x}, \tilde{x}')$
l – overall number of slave processes
$r \in [0..(l-1)]$ – my process ID
Output:
New clusters $(\tilde{x}, \tilde{x}')$
Main loop:
For each iteration $(1, \ldots, l-1)$ **do**
  Build a random permutation Ψ of all values $x \in (\tilde{x}, \tilde{x}')$
  For each $x \in \Psi$ **do**
    Move x from its cluster into another if this leads to $\Delta I(\tilde{X}; \tilde{Y}) > 0$ (from Eq. (11))
  If iteration number is odd **then**
    If $r == 0$ **then** swap clusters $\tilde{x}$ and $\tilde{x}'$
    Send cluster $\tilde{x}$ to process with ID $(r+1)\%l$
    Receive cluster $\tilde{x}$ from process with ID $(l+r-1)\%l$
  Else
    Send cluster $\tilde{x}'$ to process with ID $(l+r-1)\%l$
    Receive cluster $\tilde{x}'$ from process with ID $(r+1)\%l$
  Synchronize with all the other slave processes

Algorithm 2: Slave process.

pairs and sending them to the slave processes, the master node switches to the wait state, while the slave processes work autonomously, communicating with each other. Each slave process receives two clusters and shuffles them while optimizing the objective. After the shuffling task is completed, the slave is ready to send and receive clusters. It is enough to send (and receive) only *one* cluster of each pair—by which the communication cost is kept at its minimum.

Figure 3 illustrates our communication protocol, where (for simplicity) we assume that k is even. As each slave process holds two clusters at each time point, we can enumerate the "seats" for clusters as shown in the figure, where the upper row defines process numbers at the same time. The protocol is to alternate sending the upper of the clusters to the right and then the lower one to the left. So in general, each cluster will move in just one direction, and be moved every second iteration. Node number 0 is an exception, in that it keeps the cluster initially sitting in seat number $k-1$ all the time in that place, and hence always sends the other cluster. It thereby inverts the direction in which a cluster is moving. To clarify the order: without the termination criterion, a cluster starting from seat 0 would follow the cyclic seat sequence $1, 1, 2, 2, \ldots, (k/2-1), (k/2-1), 0, k/2, k/2, (k/2+1), \ldots, (k-2), (k-2), 0$. The case of an odd number of clusters is treated analogously, where the seat labeled with $k-1$ in Figure 3 is kept unoccupied.

PROPOSITION 4.1. *The DataLoom communication protocol guarantees that every pair of clusters meets exactly once.*

PROOF. Two clusters meet at a node iff the sum of their seat numbers modulo $(k-1)$ is 0. If we start with an iteration that moves clusters in the upper row to the right, then every two iterations later the new seat number of every cluster will be increased by 1 modulo $(k-1)$. This can easily be shown inductively. The sum will hence increase by 2 every two iterations. Similarly, if the sum of seat numbers (modulo $k-1$) is $(k-2)$, then the clusters will meet in the next iteration. It is easy to see that by adding 2 modulo $(k-1)$ it takes at most $(k-1)$ iterations until any two regular clusters

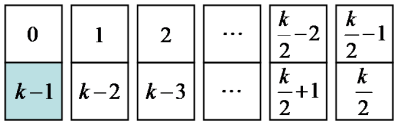

Figure 3: An illustration to the deterministic message passing algorithm.

(without the stationary one) meet, and it is also clear that every cluster will hit node 0 and meet the stationary cluster from either seat $(k/2-1)$ or $(k-2)$ when moving into the same direction for $(k-1)$ iterations. □

Together with the deterministic communication protocol, we propose a stochastic one, in which, after the cluster shuffling is completed, a slave process sends one cluster to another process chosen randomly. The exact protocol is precomputed by the master and then distributed to the slaves. It keeps track of the cluster transfers such that at each point of time each slave node has two clusters to process. The stochastic protocol overcomes the problem of the deterministic protocol, which preserves the initial ordering of clusters that may presumably be disadvantageous. However, the stochastic protocol does not provide the completeness guarantee given in Proposition 4.1.

A schematic summary of the DataLoom algorithm is given in Figure 4. The collection of slave processes operates as a loom, that uses the communication protocol as a shuttle to weave clusters. When the cluster "fabric" is woven, the master process collects all the clusters and switches to optimizing another modality. Obviously, our method can be generalized to process any number of modalities organized in an interaction graph, to be traversed by the master. The method's complexity increases only linearly with increasing the number of edges in the interaction graph.

Note that, regarding the computational complexity (without communication), DataLoom is no more expensive than parallel k-means. At each slave process, DataLoom probes whether moving a data point to the other cluster on that machine increases the objective. Totalling the number of these comparisons, we find that on average we probe each cluster exactly once per data point, so we probe as many point-cluster pairs as parallel k-means and parallel IT-CC.

5. IMPLEMENTATION AND EXPERIMENTATION

In our implementation of the DataLoom algorithm, the communication is based on the Massage Passing Interface (MPI) [32]. We decided to apply the traditional MPI instead of a currently more popular MapReduce scheme because an iterative application of MapReduce has a substantial disadvantage: backpropagating the data from a reducer to the next mapper requires a disk access, which is very expensive in our setup. The DataLoom algorithm is deployed on a Hewlett Packard XC Linux cluster system that consists of 62 eight-core machines with 16Gb RAM each.

As a baseline for our large-scale experiments, together with the parallelized IT-CC algorithm, we used a parallelized version of the double k-means algorithm (see, e.g. [28]). Double k-means is basically the IT-CC optimization procedure that minimizes the traditional k-means objective function (the sum of Euclidian distances of data points to

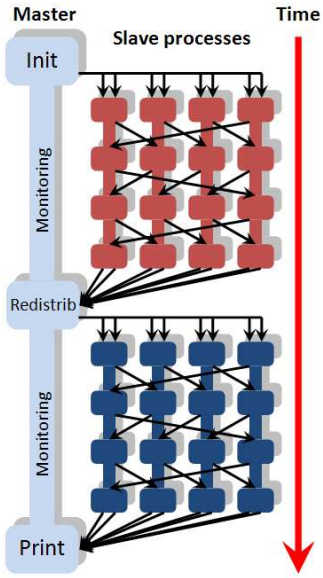

Figure 4: The DataLoom algorithm.

their centroids). We parallelized it analogously to the IT-CC parallelization (see Section 3.2).

5.1 Comparison with sequential co-clustering

Our first objective is to show that the performance of the DataLoom algorithm is comparable to the one of its sequential ancestor. To meet this objective, we replicate the experimental setup of Bekkerman et al. [1], who test ITC algorithms on six relatively small textual datasets. Our evaluation measure is *micro-averaged accuracy* that is defined as follows. Let $\mathcal{C}$ be the set of ground truth categories. For each cluster $\tilde{x}$, let $\mu_{\mathcal{C}}(\tilde{x})$ be the maximal number of elements of $\tilde{x}$ that belong to one category. Then, the precision of $\tilde{x}$ with respect to $\mathcal{C}$ is defined as $Prec(\tilde{x},\mathcal{C}) = \mu_{\mathcal{C}}(\tilde{x})/|\tilde{x}|$. The micro-averaged precision of the entire clustering $\tilde{X}$ is:

$$Prec(\tilde{X},\mathcal{C}) = \sum_{\tilde{x}} \mu_{\mathcal{C}}(\tilde{x}) / \sum_{\tilde{x}} |\tilde{x}|, \qquad (12)$$

which is the portion of data points that belong to dominant categories. If the number of clusters is equal to the number of categories, then $Prec(\tilde{X},\mathcal{C})$ equals micro-averaged recall and thus equals clustering accuracy.

For simplicity, we choose four out of the six datasets used by Bekkerman et al. [1]—the ones that have an even number of categories. Three of those datasets (*acheyer*, *mgondek*, and *sanders-r*) are small collections of 664, 297, and 1188 email messages, grouped into 38, 14, and 30 folders, respectively. The fourth dataset is the widely-used benchmark 20 Newsgroups (20NG) dataset, that consists of 19,997 postings submitted to 20 newsgroups. About 4.5% of the 20NG documents are duplications—we do not remove them, for better replicability. For all the four datasets, we simultaneously cluster documents and their words. For email datasets, we also cluster the third modality, which is the names of email correspondents. For the 3-way clustering, we use our CWO optimization scheme (see Section 2).

The summary of our results is given in Table 1. Besides comparing to SCC and IT-CC, we compared DataLoom

Table 1: Clustering accuracy on small datasets. The standard error of the mean is given after the ± sign.

Dataset	k-means	LDA	IT-CC	SCC	2way DataLoom (deterministic)	2way DataLoom (stochastic)	3way DataLoom (stochastic)
acheyer	24.7	44.3 ± 0.4	39.0±0.6	46.1±0.3	43.7±0.5	42.4±0.5	46.7±0.3
mgondek	37.0	68.0 ± 0.8	61.3±1.5	63.4±1.1	63.3±1.8	64.6±1.2	73.8±1.7
sanders-r	45.5	63.8 ± 0.4	56.1±0.7	60.2±0.4	59.8±0.9	61.3±0.8	66.5±0.2
20NG	16.1	56.7 ± 0.6	54.2±0.7	57.7±0.2	55.1±0.7	55.6±0.7	N/A

against the standard uni-modal k-means, as well as against *Latent Dirichlet Allocation (LDA)* [5]—a popular generative model for representing document collections. In LDA, each document is represented as a distribution of topics, and parameters of those distributions are learned from the data. Documents are then clustered based on their posterior distributions (given the topics). We used Xuerui Wang's LDA implementation [25] that applies Gibbs sampling with 10000 sampling iterations.

As we can see in the table, the empirical results approve our theoretical argumentation from Section 3.3—sequential co-clustering significantly outperforms the IT-CC algorithm. Our 2-way parallelized algorithm demonstrates very reasonable performance: only in two of the four cases it is inferior to the SCC. It is highly notable that our 3-way DataLoom algorithm achieves the best results, outperforming by more than 5% (on the absolute scale) all its competitors on *mgondek*. When comparing the deterministic and stochastic communication protocols, we notice that they perform comparably. For the rest of our experiments, we use the stochastic version.

5.2 The RCV1 dataset

The RCV1 [22] dataset is by far the largest fully labeled text categorization dataset available to the research community. It consists of 806,791 documents each of which belongs to a hierarchy of categories. The top level of the hierarchy contains only four categories, while the second level contains 55 categories. In our experiment, we ignore the top level and map categories from all the lower levels onto their parents from the second level (using this scheme, 27076 documents are not assigned into any category, and therefore are always considered as wrongly categorized). We remove stopwords and low frequency words (leaving 150,032 distinct words overall). Represented as a contingency table, the resulting data contains over 120 billion entries. We are aware of only one previous work [29] where the entire RCV1 collection was clustered. Following this work, we use the *clustering precision* measure, given in Equation (12). We built 800 document clusters and 800 word clusters. We plot the precision over the clustering iterations and compare DataLoom with the parallelized IT-CC, as well as with parallelized double k-means. The results are presented in Figure 5 (left), where DataLoom has a clear advantage over the other methods. We also plot the mutual information $I(\tilde{X};\tilde{Y})$ after each iteration, and show that DataLoom is able to construct clusterings with 20% higher mutual information.

5.3 The Netflix dataset

Another data set we used in our experiments was taken from the Netflix challenge. It contains the ratings of 17,770 movies given by 480,189 users. We did not consider the actual value of ratings, but wanted to predict for a number of given user-movie pairs whether or not this user rated this movie. This resembles one of the tasks of KDD'07 Cup, and we used the evaluation set provided as part of that Cup. We built 800 user clusters and 800 movie clusters.

Our prediction method is directly based on the the natural approximation q (defined in Equation (4)) of our (normalized) boolean movie-user rating matrix p. The quality of this approximation is prescribed by the quality of the co-clustering. The intuition behind our experiment is that capturing more of the structure underlying this data helps in better approximating the original matrix. We ranked all the movie-user pairs in the hold-out set with respect to the predicted probability of q. Then we computed the Area Under the ROC Curves (AUC) for the three co-clustering algorithms. To establish a lower bound, we also ranked the movie-user pairs based on the pure popularity score $p(x)p(y)$. The results are shown in Figure 5 (right). In addition to that, as in the RCV1 case, we directly compared the objective function values of the co-clusterings produced by DataLoom and IT-CC. As for RCV1, DataLoom shows an impressive advantage compared to the other methods.

6. CONCLUSION

This paper comes as an attempt to dramatically scale up a strong data clustering method, while applying parallelization. The resulting algorithm is applied to two large labeled data corpora, RCV1 and Netflix, of hundreds of thousands data instances each. The algorithm is, by all means, applicable to datasets orders of magnitude larger than that, but we decided on these two datasets for the evaluation purposes only.

As far as the speedup is concerned, on small datasets (see Section 5.1) the DataLoom method is not gaining particularly impressive advantage over non-parallelized methods. Naturally, small datasets can be clustered using sequential clustering as is. On large datasets, however, the parallelization is vital. Basically, SCC is not applicable to the large datasets: on RCV1, for example, it would have run for months (assuming that it can fit the RAM). Thus, applying the data weaving parallelization makes real what would have been infeasible otherwise.

Our immediate future work goal is to apply DataLoom to large-scale multi-modal data with more than two modalities. A possible candidate is the Netflix data, where the movies are associated with their textual descriptions (the third modality), and the full cast (the fourth modality). Extensions of our method to constrained co-clustering [27] and to relational clustering [24] are also being considered.

7. ACKNOWLEDGEMENTS

We would like to thank Eric Wu and N. K. Krishnan for their excellent technical support of the XC cluster.

8. REFERENCES

[1] R. Bekkerman, R. El-Yaniv, and A. McCallum. Multi-way distributional clustering via pairwise interactions. In *Proceedings of ICML-22*, pages 41–48, 2005.

[2] R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter. On feature distributional clustering for text categorization. In *Proceedings of SIGIR*, pages 146–153, 2001.

[3] R. Bekkerman, M. Sahami, and E. Learned-Miller. Combinatorial Markov Random Fields. In *Proceedings of ECML-17*, 2006.

[4] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, 48(3), 1986.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[6] C. Böhm, C. Faloutsos, J.-Y. Pan, and C. Plant. Robust information-theoretic clustering. In *Proceedings of ACM SIGKDD*, pages 65–75, 2006.

[7] S. Brecheisen, H.-P. Kriegel, and M. Pfeifle. Parallel density-based clustering of complex objects. In *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2006.

[8] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. R. Bradski, A. Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

[9] K. Crammer, P. Talukdar, and F. Pereira. A rate-distortion one-class model and its applications to clustering. In *Proceedings of the 25st International Conference on Machine Learning*, 2008.

[10] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Symposium on Operating System Design and Implementation (OSDI)*, pages 137–150, 2004.

[11] D. Deb and R. A. Angryk. Distributed document clustering using word-clusters. In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 376–383, 2007.

[12] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of SIGKDD-9*, pages 89–98, 2003.

[13] I. S. Dhillon and D. S. Modha. A data clustering algorithm on distributed memory multiprocessors. In *Large-Scale Parallel Data Mining*, volume 1759 of *Lecture Notes in Artificial Intelligence*, 2000.

[14] R. El-Yaniv and O. Souroujon. Iterative double clustering for unsupervised and semi-supervised learning. In *Advances in Neural Information Processing Systems (NIPS-14)*, 2001.

[15] G. Forman and B. Zhang. Distributed data clustering can be efficient and exact. *SIGKDD Exploration Newsletter*, 2(2):34–38, 2000.

[16] N. Friedman, O. Mosenzon, N. Slonim, and N. Tishby. Multivariate information bottleneck. In *Proceedings of UAI-17*, 2001.

[17] B. Gao, T.-Y. Liu, X. Zheng, Q.-S. Cheng, and W.-Y. Ma. Consistent bipartite graph co-partitioning for star-structured high-order heterogeneous data co-clustering. In *Proceedings of ACM SIGKDD*, 2005.

[18] P. E. Hadjidoukas and L. Amsaleg. Parallelization of a hierarchical data clustering algorithm using openmp. In *Proceedings of the International Workshop on OpenMP (IWOMP)*, Reims, France, June 2006.

[19] M. Johnson, R. H. Liao, A. Rasmussen, R. Sridharan, D. Garcia, and B. Harvey. Infusing parallelism into introductory computer science using mapreduce. In *Proceedings of SIGCSE: Symposium on Computer Science Education*, 2008.

[20] D. Judd, P. K. McKinley, and A. K. Jain. Large-scale parallel data clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):871–876, 1998.

[21] S. L. Lauritzen. *Graphical Models*. Clarendon Press, 1996.

[22] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397, 2004.

[23] K. Liu, H. Kargupta, J. Ryan, and K. Bhaduri. Distributed data mining bibliography. http://www.cs.umbc.edu/∼hillol/DDMBIB, 2004.

[24] B. Long, Z. Zhang, and P. S. Yu. A probabilistic framework for relational clustering. In *Proceedings of the ACM SIGKDD*, pages 470–479, 2007.

[25] A. McCallum, A. Corrada-Emmanuel, and X. Wang. Topic and role discovery in social networks. In *Proceedings of IJCAI-19*, pages 786–791, 2005.

[26] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of ACM SIGKDD*, pages 169–178, 2000.

[27] R. Pensa and J.-F. Boulicaut. Constrained co-clustering of gene expression data. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 25–36, 2008.

[28] R. Rocci and M. Vichi. Two-mode multi-partitioning. *Computational Statistics and Data Analysis*, 52(4), 2008.

[29] N. Rooney, D. Patterson, M. Galushka, and V. Dobrynin. A scaleable document clustering approach for large document corpora. *Information Processing and Management*, 42(5):1163–1175, 2006.

[30] N. Slonim, N. Friedman, and N. Tishby. Unsupervised document classification using sequential information maximization. In *Proceedings of SIGIR-25*, 2002.

[31] N. Slonim and N. Tishby. Agglomerative information bottleneck. In *Advances in Neural Information Processing Systems 12 (NIPS)*, pages 617–623, 2000.

[32] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI—The Complete Reference: Volume 1, The MPI Core*. MIT Press, 2nd edition, 1998.

[33] C. Sutton and A. McCallum. Piecewise training of undirected models. In *Proceedings of UAI-21*, 2005.

[34] N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method, 1999. Invited paper to the 37th Annual Allerton Conference on Communication, Control, and Computing.

[35] X. Xu, J. Jäger, and H.-P. Kriegel. A fast parallel clustering algorithm for large spatial databases. *Data Mining and Knowledge Discovery*, 3(3):263–290, 1999.

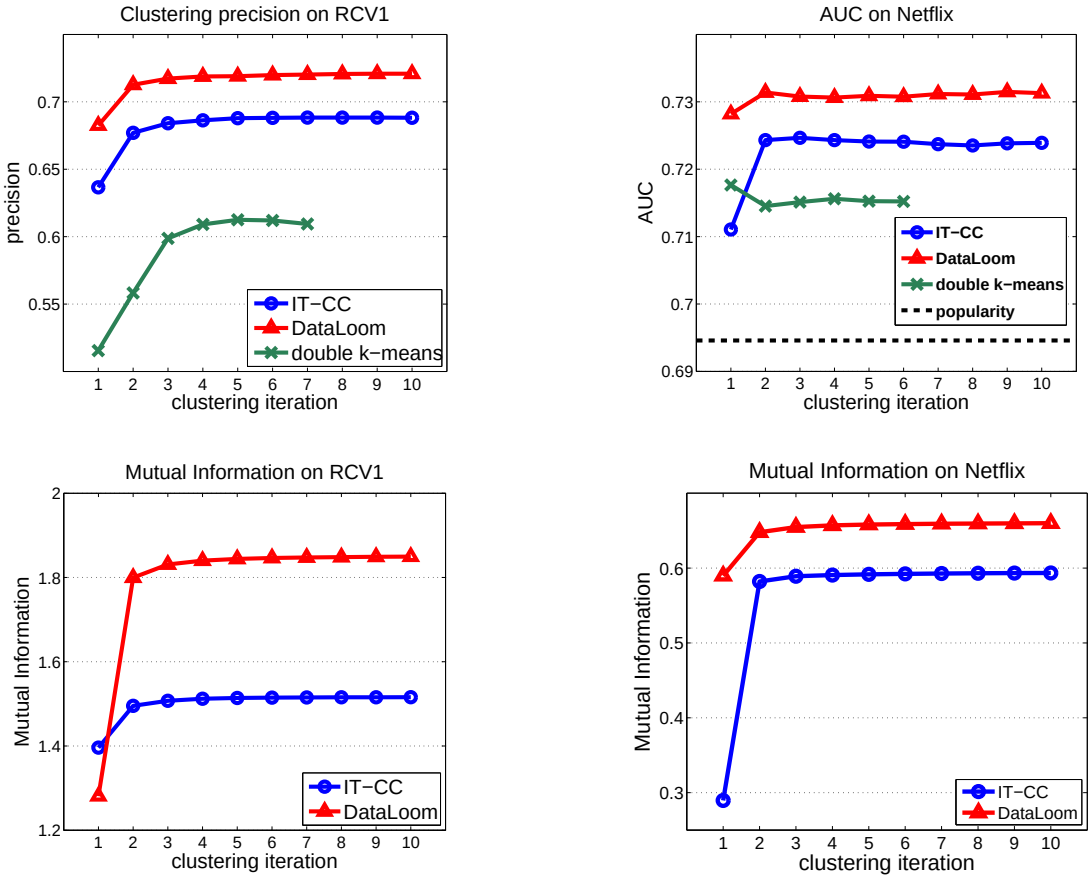

Figure 5: Clustering results on RCV1 (left) and Netflix (right)