Text Mining with Hybrid Clustering Schemes

Jacob Kogan
kogan@umbc.edu
Department of Mathematics and Statistics
UMBC, Baltimore, MD 21250
Charles Nicholas
nicholas@umbc.edu
Department of Computer Science and Electrical Engineering
UMBC, Baltimore, MD 21250
and
Vladimir Volkovich
zeev@actcom.co.il
Software Engineering Department
ORT Braude Academic College
Karmiel 21982, Israel

January 28, 2003

Abstract

Hybrid information retrieval (IR) schemes combine different normalization techniques and similarity functions. Hybrid schemes provide an efficient technique to improve precision and recall (see e.g., [4]). This paper reports a hybrid clustering scheme that applies a singular value decomposition (SVD) based algorithm followed by a k–means type clustering algorithm. The output of the first algorithm becomes the input of the next one. The second algorithm generates the final partition of the data set. We report results of numerical experiments performed with three k–means type clustering algorithms. Those are: the classical k–means (see e.g., [9]), the spherical k–means (see [7]), and the information–theoretical clustering algorithm introduced recently by [8], and [1]. A comparison with the results reported by [7] is provided.

1. Introduction

Clustering is an important, old and very difficult problem. While a wide variety of clustering algorithms are already available in the literature in many cases a single algorithm is applied to handle a given data set.

We advocate a multistage approach to clustering. We suggest a sequence of two algorithms to be applied to a data set. The outcome of the first algorithm becomes input to the second one. The final partition is generated by the second algorithm. We call a sequence of two (or more) clustering algorithms applied to a data set a *hybrid scheme*.

The second stage of the hybrid scheme described in the paper is a k-means type algorithm. A number of algorithms similar to the classical k-means algorithm [9] are available in the literature. Motivated by text mining applications, we focus on the following three algorithms: a) the classical k-means, b) the spherical k-means [7], and c) the information theoretical k-means (IT–means) [8]. While the classical k-means algorithm is capable of handling general vector sets, the spherical k–means is designed to cluster unit l_2 vectors, and the IT-means to cluster unit l_1 vectors.

The k–means type algorithms are known to be sensitive to the choice of initial partition. To generate a good initial partition for the second step of the hybrid scheme we first apply the Spherical Principal Directions Divisive Partitioning (sPDDP) algorithm (see [6]). The SVD type algorithm approximates the unit l_2 norm document vectors by vectors on a one dimensional circle. The circle is the intersection of the unit sphere and the best 2 dimensional approximation of the data set. The algorithm is motivated by the Principal Direction Divisive Partitioning algorithm introduced by Boley [3].

The outline of the paper is the following. In Section 2 we briefly describe the clustering algorithms. Section 3 describes the data set and a feature selection technique we use in this paper. Section 4 contains results of numerical experiments. Brief conclusions and further research directions are outlined in Section 5.

2. Clustering algorithms

We start with a brief description of the first step of the hybrid scheme (for details we refer the reader to [6]).

2.1. SVD based algorithms

Clustering of a vector set is, in general, a difficult task. There is, however, an exception. When all the vectors belong to a one dimensional line, clustering becomes relatively easy. In many cases a good partition of a one–dimensional set Y into two subsets Y_1 and Y_2 amounts to a selection of a number, say μ, so that

$$Y_1 = \{y \ : \ y \in Y, \ y \leq \mu\}, \text{ and } Y_2 = \{y \ : \ y \in Y, \ y > \mu\}$$

The Principal Direction Divisive Partitioning (PDDP) clustering algorithm introduced by D. Boley [3] approximates a set of vectors by a line defined by a singular vector of the sparse "term by document" matrix. PDDP projects the vectors on the line, partitions the projections into two clusters, and builds a two cluster partition of the original vector set from the one dimensional "approximation" clusters. The algorithm then recursively partitions each cluster separately until a stopping criterion is met.

PDDP is a clustering algorithm capable of handling a general vector set. Normalized "document vectors" reside on a unit sphere and may be approximated by vectors on a one dimensional circle. The circle is defined by the intersection of the sphere and a two dimensional plane defined by two singular vectors of the "term by document" matrix. An efficient partitioning of vectors on a one dimensional circle is provided by the spherical k–means algorithm (see [6]). The partitioning of the original vector set is induced by the partitioning of the "approximation vectors" on the one dimensional circle. Like PDDP, the algorithm proceeds recursively until a stopping criterion is met.

The description of the algorithm (which we call the Spherical Principal Directions Divisive Partitioning, or sPDDP) and clustering results are provided in [6], where it is shown that sPDDP significantly outperforms PDDP on the data set described in Section 3.

Partitions generated by sPDDP are fed as input to k–means like algorithms described below.

2.2. k–means type algorithms

The following is a basic description of a k–means type clustering algorithm:

Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ be a set of vectors in a subset $\mathbf{X}$ of the n-dimensional Euclidean space $\mathbf{R}^n$. Consider a partition $\Pi = \{\pi_1, \ldots, \pi_k\}$ of the set, i.e.,

$$\pi_1 \cup \ldots \cup \pi_k = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}, \text{ and } \pi_i \bigcap \pi_j = \emptyset \text{ if } i \neq j.$$

Given a real valued "quality function" q whose domain is the set of subsets of $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ the quality of the partition is given by $\mathcal{Q}(\Pi) = q(\pi_1) + \ldots + q(\pi_k)$. The problem is to identify an optimal partition $\{\pi_1^o, \ldots, \pi_k^o\}$, i.e., one that optimizes $q(\pi_1) + \ldots + q(\pi_k)$. Often the function q is associated with a "dissimilarity measure", or a *distance-like* function $d(\mathbf{x}, \mathbf{y})$ that satisfies the following basic properties:

$$d(\mathbf{x}, \mathbf{y}) \geq 0, \ \forall (\mathbf{x}, \mathbf{y}) \in \mathbf{X} \ \text{ and } \ d(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y}$$

(we will call $d(\cdot, \cdot)$ a *distance-like* function, since we do not require d to be either symmetric or to satisfy the triangle inequality). To describe the relation between q and d we define a centroid $\mathbf{c}$ of a cluster π by

$$\mathbf{c} = \mathbf{c}(\pi) = \arg\operatorname{opt}\left\{\sum_{\mathbf{x} \in \pi} d(\mathbf{y}, \mathbf{x}), \ \mathbf{y} \in \mathbf{X}\right\}, \tag{2.1}$$

where by $\arg\operatorname{opt} f(x)$ we denote a point x_0 where the function f is optimized. Depending on the choice of d, "opt" is either "min" or "max." Indeed, to define a centroid one would like to *minimize* (2.1) with $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2$. On the other hand one would like to *maximize* the same expression when $d(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}$, and all the vectors are normalized. To simplify the exposition in what follows we shall primarily address the case "opt= min" keeping in mind that the inequalities should be reversed in the other case. If $q(\pi)$ is defined as $\sum_{\mathbf{x} \in \pi} d(\mathbf{c}(\pi), \mathbf{x})$, then centroids and partitions are associated as follows:

1. For each set of k centroids $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ one can define a partition $\{\pi_1, \ldots, \pi_k\}$ of the set $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ by:

$$\pi_i = \{\mathbf{x}_j \ : \ d(\mathbf{c}_i, \mathbf{x}_j) \leq d(\mathbf{c}_l, \mathbf{x}_j) \text{ for each } l \neq i\} \tag{2.2}$$

   (we break ties arbitrarily).

2. Given a partition $\{\pi_1, \ldots, \pi_k\}$ of the set $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ one can define the corresponding centroids $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ by:

$$\mathbf{c}_i = \arg\min\left\{\sum_{\mathbf{x} \in \pi_i} d(\mathbf{y}, \mathbf{x}), \ \mathbf{y} \in \mathbf{X}\right\}, \tag{2.3}$$

A batch k–means type algorithm is the following procedure:

Algorithm 2.1 Batch k–means like algorithm
Given a user supplied tolerance $\mathtt{tol_B} > 0$ *do the following:*

```
1. Set $t = 0$.
2. Start with an initial partitioning $\Pi^t = \left\{\pi_1^t, \ldots, \pi_k^t\right\}$
3. Apply (2.3) to recompute centroids.
4. Apply (2.2) to compute the partition $\mathtt{nextKM}\left(\Pi^t\right) = \Pi^{(t+1)} = \left\{\pi_1^{(t+1)}, \ldots, \pi_k^{(t+1)}\right\}$.
5. If $\mathcal{Q}\left(\Pi^t\right) - \mathcal{Q}\left(\mathtt{nextKM}\left(\Pi^t\right)\right) < \mathtt{tol_B}$
        set $t = t + 1$
        goto Step 3
6. Stop.
```

We now pause briefly to display a number of examples of *distance-like* functions $d(\mathbf{x}, \mathbf{y})$, data domains $\mathbf{X}$, norm constraints imposed on the data, k–means like algorithms, and centroid formulas (see Table 1 below). For a set of vectors $\pi = \{\mathbf{y}_1, \ldots, \mathbf{y}_p\}$ we denote the mean of the set by $\mathbf{m}(\pi)$, i.e., $\mathbf{m}(\pi) = \dfrac{\mathbf{y}_1 + \ldots + \mathbf{y}_p}{p}$. We denote by $\mathbf{S}_2^{n-1}$ the l_2 unit sphere in $\mathbf{R}^n$, i.e.

$$\mathbf{S}_2^{n-1} = \left\{\mathbf{x} \ : \ \mathbf{x} \in \mathbf{R}^n, \ (\mathbf{x}[1])^2 + \ldots + (\mathbf{x}[n])^2 = 1\right\},$$

and $\mathbf{S}_1^{n-1}$ stands for the l_1 unit sphere in $\mathbf{R}^n$, i.e.

$$\mathbf{S}_1^{n-1} = \{\mathbf{x} \ : \ \mathbf{x} \in \mathbf{R}^n, \ |\mathbf{x}[1]| + \ldots + |\mathbf{x}[n]| = 1\},$$

here $\mathbf{x} = (\mathbf{x}[1], \ldots, \mathbf{x}[n])^T \in \mathbf{R}^n$.

While computationally efficient the batch k–means often gets trapped at a local minimum even for simple one dimensional data sets [6]. An incremental version of k–means may (at least partially) remedy the problem. To describe the algorithm we need additional definitions.

Definition 2.1 *A first variation of a partition* $\Pi = \{\pi_1, \ldots, \pi_k\}$ *is a partition* $\Pi' = \{\pi'_1, \ldots, \pi'_k\}$ *obtained from* Π *by removing a single vector* $\mathbf{x}$ *from a cluster* π_i *of* Π *and assigning this vector to an existing cluster* π_j *of* Π.

$d(\mathbf{y},\mathbf{x})$	$\mathbf{X}$	constraint	"opt"	Algorithm 2.1 becomes	$\mathbf{c}(\pi)$
$\|\|\mathbf{x}-\mathbf{y}\|\|^2$	$\mathbf{R}^n$	none	min	classical batch k–means (see [9])	$\mathbf{m}(\pi)$
$\mathbf{x}^T\mathbf{y}$	$\mathbf{S}_2^{n-1}\bigcap\mathbf{R}_+^n$	$\|\|\mathbf{x}\|\|_2=1$ and $\mathbf{x}[i]\geq 0$	max	spherical batch k–means (see [7])	$\frac{\mathbf{m}(\pi)}{\|\|\mathbf{m}(\pi)\|\|_2}$
$\sum_{i=1}^n \mathbf{x}[i]\log\frac{\mathbf{x}[i]}{\mathbf{y}[i]}$	$\mathbf{S}_1^{n-1}\bigcap\mathbf{R}_+^n$	$\|\|\mathbf{x}\|\|_1=1$ and $\mathbf{x}[i]\geq 0$	min	batch IT – means (see [8])	$\mathbf{m}(\pi)$

Table 1: examples of three clustering algorithms

Note that the partition Π is a first variation of itself. Next we look for the "steepest descent" first variation, i.e., a first variation that leads to the maximal decrease of the objective function. The formal definition follows:

Definition 2.2 *The partition* $\texttt{nextFV}(\Pi)$ *is a first variation of a partition* Π *if for each first variation* Π' *one has*

$$\mathcal{Q}\left(\texttt{nextFV}(\Pi)\right) \leq \mathcal{Q}\left(\Pi'\right). \tag{2.4}$$

To benefit from the speed of batch iterations and the accuracy of the incremental iterations we apply them in "tandem" as follows.

Algorithm 2.2 Batch-incremental tandem algorithm.
For user supplied tolerances $\texttt{tol}_\texttt{B} > 0$ *and* $\texttt{tol}_\texttt{I} > 0$ *do the following:*

```
1. Set the index of iteration $t = 0$.
2. Start with an arbitrary partitioning $\Pi^t = \left\{\pi_1^t, \dots, \pi_k^t\right\}$.
3. Generate the partition $\Pi^{(t+1)} = \texttt{nextKM}\left(\Pi^t\right)$.
   if $\left[\mathcal{Q}\left(\Pi^t\right) - \mathcal{Q}\left(\texttt{nextKM}\left(\Pi^t\right)\right) < \texttt{tol}_\texttt{B}\right]$
       set $t = t + 1$
       goto Step 3
```

```
4. Generate the partition $\Pi^{(t+1)} = \texttt{nextFV}\left(\Pi^t\right)$.
   if $\left[\mathcal{Q}\left(\Pi^t\right) - \mathcal{Q}\left(\texttt{nextFV}\left(\Pi^t\right)\right) < \texttt{tol}_\texttt{I}\right]$
        set $t = t + 1$
        goto Step 3

5. Stop.
```

Assessment of the efficiency and computational complexity of Step 4 of Algorithm 2.2 for the classical k–means algorithm as well as the spherical k–means algorithm have been reported in [10] and [5] respectively. The addition of this step always leads to improvement of clustering results, and comes at virtually no additional computational expense.

Computational complexity analysis for the incremental version of the Information–Theoretical k-means algorithm is provided in [1]. Computational cost of Step 4 is of the same order as the computational cost of the batch version of the algorithm (i.e., the tandem version of the Information-Theoretical k-means algorithm is twice as expensive as its batch version). We shall further discuss efficiency and computational complexity of the algorithms in Section 4.

3. Text data and feature selection

Dhillon and Modha [7] have recently used the spherical k–means algorithm for clustering text data. In one of their experiments the algorithm was applied to a data set containing 3893 documents [7]. The data set contains the following three document collections (available from `ftp://ftp.cs.cornell.edu/pub/smart`):

- DC0 (Medlars Collection, 1033 medical abstracts).
- DC1 (CISI Collection, 1460 information science abstracts).
- DC2 (Cranfield Collection, 1400 aerodynamics abstracts).

Partitioning the entire collection into three clusters generates the "confusion" matrix reported in Table 2 (see [7]). The entry ij in Table 2 is the number of documents that belong to cluster i and document collection j. The "confusion" matrix shows that only 69 documents (i.e., less that 2% of the entire collection) have been "misclassified" by the algorithm. After removing stopwords Dhillon and Modha [7] reported 24,574 unique

	DC0	DC1	DC2
cluster 0	1004	5	4
cluster 1	18	1440	16
cluster 2	11	15	1380

Table 2: spherical k–means generated "confusion" matrix with **69** "misclassified" documents using 4099 terms

words, and after eliminating low–frequency and high–frequency words they selected **4,099** words to construct the vector space model.

Our data set is a merger of the same three document collections obtained from `http://www.cs.utk.edu/~lsi/`. The Cranfield collection used by Dhillon and Modha contained two empty documents. These two documents have been removed from DC2. The other document collections are identical. In what follows we ignore the difference and keep notations DC0, DC1, and DC2.

We denote the overall collection of 3891 documents by DC. After stopword removal (see `ftp://ftp.cs.cornell.edu/pub/smart/english.stop`), and Porter stemming (see [12]) the data set contains 15,864 unique terms. (No stemming was applied to the 24,574 unique words reported in [7]).

To exploit statistics of term occurrence throughout the corpus we remove terms that occur in less than r sentences across the collection, and denote the remaining terms by slice(r) (the experiments in this paper are performed with $r = 20$). The first l best quality terms that belong to slice(r) define the dimension of the vector space model.

We denote the frequency of a term $\mathbf{t}$ in the document $\mathbf{d}_j$ by f_j. Following an approach outlined in [6] we measure the quality of the term $\mathbf{t}$ by

$$\sum_{j=1}^{m} f_j^2 - \frac{1}{m}\left[\sum_{j=1}^{m} f_j\right]^2, \tag{3.1}$$

where m is the total number of documents in the collection. Note that the quality of a term is proportional to the term frequency variance.

To evaluate the impact of feature selection based on the quality function (3.1) on clustering we conducted the following experiment. The best quality **600** terms are selected, and l_2 unit norm vectors for the 3891 documents are built (we use the `tfn` scheme to construct document vectors, for details see [7]). A two step procedure is employed to partition the 3891 vectors into three clusters:

1. the sPDDP algorithm (see [6]) is applied to generate three clusters (the obtained clusters are used as an initial partition in the next step),

2. the vectors are re–normalized as needed, and the three k–means like algorithms are applied to the partition obtained in the previous step.

The results of the experiment with $\texttt{tol}_{\texttt{B}} = \texttt{tol}_{\texttt{I}} = 0$ are provided in the next section.

4. Experimental results

In what follows we display clustering results for the document collection DC described in Section 3 and compare the results with those reported by [7] and [6]. The confusion matrix for the three cluster partition generated by sPDDP is given in Table 3 below. Note that a good clustering procedure should be able to produce a "confusion" matrix

	DC0	DC1	DC2
cluster 0	1000	3	1
cluster 1	8	10	1376
cluster 2	25	1447	21
"empty" documents			
cluster 3	0	0	0

Table 3: sPDDP generated initial "confusion" matrix with **68** "misclassified" documents using best 600 terms

with a single "dominant" entry in each row. The "confusion" matrices for the three clusters provided in Tables 3 and 4, 5 and 6 illustrate this remark.

When the number of terms is relatively small, some documents may contain no selected terms, and their corresponding vectors are zeros. We always remove these vectors ahead of clustering and assign the "empty" documents into a special cluster. This cluster is the last row in the "confusion" matrix (and is empty in the experiment reported in Tables 3, 4, 5 and 6 below).

While the number of selected terms is only 600 (i.e., only about 15% of the number of terms reported in [7]) the quality of the sPDDP generated confusion matrix is comparable with that of the confusion matrix generated by the spherical k–means algorithm (see Table 2). A subsequent application of the three k–means like algorithms to the partition generated by sPDDP further improves the confusion matrix (see Tables 4, 5 and 6).

	DC0	DC1	DC2
cluster 0	1023	21	10
cluster 1	1	3	1370
cluster 2	9	1436	18
"empty" documents			
cluster 3	0	0	0

Table 4: k–means generated final "confusion" matrix with **62** "misclassified" documents.

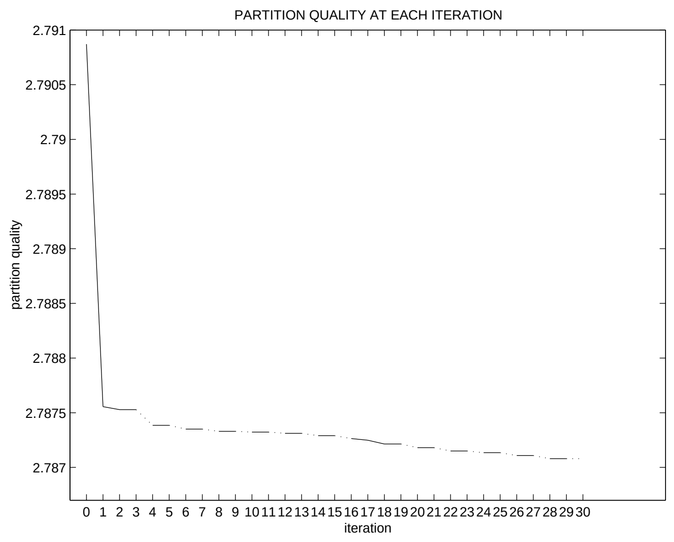
	DC0	DC1	DC2
cluster 0	1010	4	2
cluster 1	5	7	1378
cluster 2	18	1449	18
"empty" documents			
cluster 3	0	0	0

Table 5: spherical k–means generated final "confusion" matrix with **54** "misclassified" documents using best 600 terms

	DC0	DC1	DC2
cluster 0	1010	6	0
cluster 1	3	4	1387
cluster 2	20	1450	11
"empty" documents			
cluster 3	0	0	0

Table 6: IT–means generated final "confusion" matrix with **44** "misclassified" documents using best 600 terms

We pause briefly to analyze performance of the IT–means algorithm. In the experiment described above the algorithm performs 30 iterations.



The figure shows the values of $\mathcal{Q}\left(\Pi^t\right)$, $t = 0, 1, \ldots, 30$. The solid line shows changes in the objective function caused by batch IT-means iterations, and the dotted line does the same for incremental IT-means iterations. The figure shows that the lion's share of the work is done by the first three batch IT-means iterations. From iteration 4 to iteration 16 values of the objective function drop due to incremental IT-means iterations only. At iterations 17, 18 and 19 the batch IT-means kicks in. For the rest of the run the objective function changes are due to incremental IT-means iterations only.

An inspection reveals that at iteration 4 a vector $\mathbf{x}$ was moved from cluster π_3 to cluster π_2, and $\mathbf{x}$ was "missed" by the batch IT-means iteration 4 because of the infinite distance between $\mathbf{x}$ and centroids $\mathbf{c}_1$ and $\mathbf{c}_2$ of clusters π_1 and π_2 respectively (for the very same reason vectors were "missed" by the batch IT-means algorithm at iterations 5 and 19).

We remind the reader that the Information–Theoretical distance between a vector $\mathbf{y}$ and a cluster $\mathbf{c}$ is given by (see Table 1)

$$d(\mathbf{c}, \mathbf{y}) = \sum_{i=1}^{n} \mathbf{y}[i] \log \frac{\mathbf{y}[i]}{\mathbf{c}[i]}, \text{ and } d(\mathbf{c}, \mathbf{y}) = \infty \iff \exists i \text{ such that } \mathbf{y}[i] > 0 \text{ and } \mathbf{c}[i] = 0.$$

Due to condition $d(\mathbf{c}_2, \mathbf{x}) = \infty$ there is an index i such that $\mathbf{x}[i] > 0$ and $\mathbf{c}_2[i] = 0$. Since the centroid is the average of its vectors the i^{th} coordinate of each vector in cluster π_2 is 0. In the framework of the vector space model [2] this means that word i occurs in the vector document $\mathbf{x}$, and occurs in no document of the cluster π_2. The observation shows that the batch IT-means algorithm tends to generate clusters of documents with common words.

On the other hand, if the document $\mathbf{x}$ has no words at all in common with other documents in cluster π_3, but has, say, five words in common with each document in cluster π_2, then re–assignment of $\mathbf{x}$ to π_2 may improve the partitioning. While batch IT–means will not move $\mathbf{x}$ from π_3 to π_2, the incremental step of the algorithm does rectify this problem (see also [11] where the same idea is used to compute the distance between a query and a document collection). We believe the logarithmic function sensitivity to low word frequencies improves performance of the IT-means algorithm. The numerical results presented next support this claim.

Table 7 summarizes clustering results for the sPDDP algorithm and the combinations of "sPDDP+k–means", "sPDDP+spherical k–means", and "sPDDP+ IT–means" algorithms for different choices of index terms. Note that zero document vectors are created when the number of selected terms is less than 300. Table 7 indicates consistent

		documents misclassified by sPDDP			
# of terms	zero vectors	alone	+k–means	+spherical k–means	+IT–means
100	12	383	258	229	168
200	3	277	133	143	116
300	0	228	100	104	81
400	0	88	80	78	56
500	0	76	62	57	40
600	0	68	62	54	44

Table 7: Number of documents "misclassified" by sPDDP, "sPDDP+k–means", "sPDDP+spherical k–means", and "sPDDP+ IT–means".

superiority of the "sPDDP+ IT–means" algorithm.

5. Future research

We advocate an application of a two algorithms sequence for clustering a single dataset, where output of the first algorithm becomes input to the second one. Normalization of the data set is algorithm dependent. The algorithms presented operate in a relatively low dimensional vector space and produce clustering results better than or comparable with results provided in [7]. Experiments reported by [14] with 7 clustering algorithms and 12 different data sets indicate that the objective function based on cosine similarity and used in [7] "leads to the best solutions irrespective of the number of clusters for most of the data sets." Our immediate goal is to focus on the following two problems:

1. While the first step of the sequence is the SVD based sPDDP algorithm that deals with unit l_2 vectors, the second step consists of three k–means like algorithms handling either l_2 or l_1 unit vectors. Our numerical experiments indicate superiority of the IT-means algorithm that clusters l_1 unit vectors. This provides a motivation to develop a SVD based algorithm capable of handling l_1 unit vectors.

2. Motivated by success of the IT-means algorithm we plan to investigate logarithmic similarity measures for clustering text data sets. The ability to provide fast computationally efficient algorithms for computing centroids (see equation (2.3)) is crucial to the implementation of clustering algorithms. The problem has been addressed in the optimization literature (see e.g., [13] and references therein).

We plan to run the experiments on a variety of large document collections.

Acknowledgments. The work of Kogan and Nicholas was supported in part by the US Department of Defense.

References

[1] P. Berkhin and Becher J. D. Learning simple relations: Theory and applications. In *Proc. Second SIAM International Conference on Data Mining*, pages 420–436, Arlington, April 2002.

[2] M. Berry and M. Browne. *Understanding Search Engines.* SIAM, 1999.

[3] D. L. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[4] M. Cornelson, E. Greengrass, R. L. Grossman, R. Karidi, and D. Shnidman. Combining information retrieval algorithms using machine learning. In *Proceedings of the SIAM KDD Workshop on Text Mining*, to appear.

[5] I. S. Dhillon, Y. Guan, and J. Kogan. Refining clusters in high-dimensional text data. In I. S. Dhillon and J. Kogan, editors, *Proceedings of the Workshop on*

Clustering High Dimensional Data and its Applications at the Second SIAM International Conference on Data Mining, pages 71–82. SIAM, 2002.

[6] I. S. Dhillon, J. Kogan, and C. Nicholas. Feature selection and document clustering. In M.W. Berry, editor, *A Comprehensive Survey of Text Mining.* Springer-Verlag, 2003, to appear.

[7] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, January 2001. Also appears as IBM Research Report RJ 10147, July 1999.

[8] Inderjit S. Dhillon, Subramanyam Mallela, and Rahul Kumar. Enhanced word clustering for hierarchical text classification. In *Proceedings of the The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2002)*, pages 191–200, 2002.

[9] E. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21(3):768, 1965.

[10] J. Kogan. Means clustering for text data. In M.W.Berry, editor, *Proceedings of the Workshop on Text Mining at the First SIAM International Conference on Data Mining*, pages 47–54, 2001.

[11] M. Larkey, L. Connell and J. Callan. Collection selection and results merging with topically organized U.S. patents and TREC data. In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM)*, pages 282–289, McLean, VA, 2000. ACM.

[12] Porter M.F. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

[13] M. Teboulle and I. Vajda. Convergence of best φ-entropy estimates. *IEEE Transactions on Information Theory*, 39:297–301, 1993.

[14] Y. Zhao and G. Karypis. Comparison of agglomerative and partitional document clustering algorithms. In I. S. Dhillon and J. Kogan, editors, *Proceedings of the Workshop on Clustering High Dimensional Data and its Applications at the Second SIAM International Conference on Data Mining*, pages 83–93. SIAM, 2002.