SAIL: Summation-based Incremental Learning for Information-Theoretic Clustering

Junjie Wu[1], Hui Xiong[2], Jian Chen[3]

[1] School of Economics and Management, Beihang University, Beijing 100083, China, wujj@buaa.edu.cn
[2] MSIS Department, Rutgers, the State University of New Jersey, USA, hxiong@rutgers.edu
[3] Research Center for Contemporary Management, Key Research Institute of Humanities and Social Sciences at Universities, Tsinghua University, Beijing 100084, China, chenj@sem.tsinghua.edu.cn

ABSTRACT

Information-theoretic clustering aims to exploit information theoretic measures as the clustering criteria. A common practice on this topic is so-called INFO-K-means, which performs K-means clustering with the KL-divergence as the proximity function. While expert efforts on INFO-K-means have shown promising results, a remaining challenge is to deal with high-dimensional sparse data. Indeed, it is possible that the centroids contain many zero-value features for high-dimensional sparse data. This leads to infinite KL-divergence values, which create a dilemma in assigning objects to the centroids during the iteration process of K-means. To meet this dilemma, in this paper, we propose a Summation-based Incremental Learning (SAIL) method for INFO-K-means clustering. Specifically, by using an equivalent objective function, SAIL replaces the computation of the KL-divergence by the computation of the Shannon entropy. This can avoid the zero-value dilemma caused by the use of the KL-divergence. Our experimental results on various real-world document data sets have shown that, with SAIL as a booster, the clustering performance of K-means can be significantly improved. Also, SAIL leads to quick convergence and a robust clustering performance on high-dimensional sparse data.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; I.5.3 [**Pattern Recognition**]: Clustering

General Terms

Algorithms, Experimentation

Keywords

Information-theoretic Clustering, K-means Distance, SAIL

KDD'08, August 24–27, 2008, Las Vegas, Nevada, USA.

1. INTRODUCTION

In recent years, we have witnessed an increased interest in information-theoretic clustering [6, 7, 21, 16, 17, 8], since information theory [3] can be naturally used as the guidance for the clustering process. For instance, the clustering analysis can be treated as the iteration process of finding a best partition on data in a way such that the loss of mutual information due to the partitioning is the least [6]. Indeed, many information theoretic measures, such as the KL divergence and the Shannon entropy, have been widely exploited as the clustering criteria for information-theoretic clustering.

This paper is focused on the problem of K-means clustering with the KL-divergence as the proximity function. To facilitate the discussion, we call the K-means with information-theoretic measures as INFO-K-means. To better understand the theoretic foundation of INFO-K-means, we present an organized study of two different views on the objective functions of INFO-K-means. First, we derive the objective function of INFO-K-means from a probabilistic view. In this regard, we know that the probabilistic view takes several assumptions on data distributions, and the goal of INFO-K-means is to maximize the likelihood function on multinomial distributions. In contrast, the information-theoretic view has no prior assumption on data distributions. In this case, the objective function of INFO-K-means is to find a best partition on data so that the loss of mutual information is minimized. The above indicates that the information-theoretic view on INFO-K-means is more appealing, since we do not need to make any assumption on data distributions. As a result, in this paper, we take the information-theoretic view on INFO-K-means.

While INFO-K-means has the sound theoretic foundation, there are some challenging issues with INFO-K-means. For example, people have shown that, for document clustering, the performance of K-means with the KL-divergence is not better than that of the spherical K-means, which has the cosine similarity as the proximity function [21]. Indeed, for high-dimensional sparse document data, K-means with the KL-divergence often has some difficult scenarios. As an example, the centroids in sparse data usually contain many zero-value features. This creates infinite KL-divergence values, which lead to a challenge in assigning objects to the centroids during the iteration process of K-means.

A traditional way to handle this zero-value dilemma is to smooth the sparse data by adding a very small value to every instance in the data [16]. In this way, there is no instance having zero feature values. This smoothing method can avoid the zero-value dilemma for K-means with the KL-divergence, however it can degrade the clustering performance since the real data values have been changed.

As an alternative to the smoothing method, in this paper, we propose a Summation-based Incremental Learning (SAIL) method for INFO-K-means clustering. Specifically, by using an equivalent objective function, SAIL replaces the computation of the KL-divergence by the computation of the Shannon entropy. This can avoid the zero-value dilemma caused by the use of the KL-divergence. Our experimental results on various real-world document data sets have shown that, with SAIL as a booster, the clustering performance of K-means can be significantly improved. Also, SAIL leads to quick convergence and a robust clustering performance on high-dimensional sparse data. Finally, the key ideas developed in the SAIL method are not limited to the KL-divergence. These ideas can be extended to some other types of proximity functions that fit K-means clustering in the case that the values of proximity functions cannot be directly computed for data with some special properties.

2. AN OVERVIEW OF INFO-K-MEANS

To better understand the theoretic foundation of INFO-K-means, in this section, we provide an organized study of two different views, the probabilistic view and the information-theoretic view, on the objective functions of INFO-K-means.

2.1 The Objective Function of Info-K-means

K-means [13] is a prototype-based, simple partitional clustering technique which attempts to find the user-specified K clusters. These clusters are represented by their centroids (a cluster centroid is typically the mean of the data objects in that cluster). K-means has an objective function:

$$obj: \quad \min \sum_{i=1}^{K} \sum_{x \in c_i} \pi_x dist(x, m_i), \tag{1}$$

where c_i denotes cluster i, m_i is the centroid of c_i, $dist(\cdot)$ is the distance function, and π_x is the weight of x given $\sum_x \pi_x = 1$. To reach the minimum, K-means employs a heuristic clustering process as follows. First, K initial centroids are selected. Then the two-phase iterations are launched. In the first phase, every point in the data is assigned to the closest centroid, and each collection of points assigned to a centroid forms a cluster; then in the second phase, the centroid of each cluster is updated based on the points assigned to the cluster. This process is repeated until no point changes clusters.

As we know, different distance functions lead to different types of K-means. Our focus in this paper is on INFO-K-means. Let $D(x\|y)$ denote the KL-divergence between two discrete distributions x and y, we have

$$D(x\|y) = \sum_i x_i \log \frac{x_i}{y_i}. \tag{2}$$

It is easy to observe that, in most cases, $D(x\|y) \neq D(y\|x)$, so $D(\cdot)$ is not a true metric which holds the symmetric property, such as the well-known Euclidean distance. If we let $dist(\cdot) \equiv D(\cdot)$ in Formula (1), we have the objective function of INFO-K-means with the KL-divergence as follows.

$$obj: \quad \min \sum_k \sum_{x \in c_k} \pi_x D(x\|m_k), \tag{3}$$

where each instance x has been normalized before clustering. To further understand INFO-K-means, we take two different views on the objective function (Equation (3)) of INFO-K-means in the following two subsections.

2.2 The Probabilistic View

In this subsection, we first derive the objective function of INFO-K-means from a probabilistic view. Specifically, the objective function can be derived by maximizing the "partitioned" likelihood function of the EM algorithm, i.e., the crisp version of EM [21].

Let us take the document data set as the example. Assume that we have a document data set $\mathbb{D}$, which consists of K crisp partitions in multinomial distributions with different parameters, i.e., $\theta_1, \cdots, \theta_K$ respectively. Let $n(x, y)$ denote the number of occurrences of term y in document x, and $n(x) = \sum_y n(x, y)$. Then we have

THEOREM 1. *Let $L = P(X|\Theta) = \Pi_x p(x|\Theta)$ be the likelihood function, $A = -\sum_x n(x) H(p(Y|x))$, $B = \sum_x n(x)$, where $H(\cdot)$ is the Shannon entropy [3]. Then, we have*

$$\frac{A - \log L}{B} = \sum_k \sum_{x \in c_k} p(x) D(p(Y|x) \| p(Y|\theta_k)), \tag{4}$$

where $p(x) = n(x)/\sum_x n(x)$ and $p(y|x) = n(x, y)/n(x)$.

PROOF: By definition,

$$\begin{aligned} \log L &= \sum_x \log p(x|\Theta) \stackrel{a}{=} \sum_k \sum_{x \in c_k} \log p(x|\theta_k) \\ &\stackrel{b}{=} \sum_k \sum_{x \in c_k} \sum_y n(x, y) \log(p(y|\theta_k)) \\ &= \sum_k \sum_{x \in c_k} n(x) \sum_y p(y|x) \log p(y|\theta_k), \end{aligned}$$

where "a" reflects the "crisp" property of the modified EM model, and "b" follows the multinomial distribution, i.e., $p(x|\theta) = \Pi_y p(y|\theta)^{n(x,y)}$.

Meanwhile, A can be transformed into

$$A = \sum_k \sum_{x \in c_k} n(x) \sum_y p(y|x) \log p(y|x).$$

If we substitute the transformed A and $\log L$ into the left-hand-side of Equation (4), we can easily get the right-hand-side. So, the proof is completed. □

REMARK: Let us compare Equation (4) with Equation (3). If we let $\pi_x \equiv p(x)$, $x \equiv p(Y|x)$, and $m_k \equiv p(Y|\theta_k)$, we have $obj \Leftrightarrow \min(A - \log L)/B \Leftrightarrow \max \log L$. This implies that, if we take the probabilistic view of the objective function, INFO-K-means aims to maximize the likelihood function on multinomial distributions. This probabilistic view of INFO-K-means requires a series of assumptions: $p(x) = n(x)/\sum_x n(x)$, $p(y|x) = n(x, y)/n(x)$, and $p(y|\theta_k) = (p(x)/\sum_{x \in c_k} p(x))p(y|x)$. However, in the experimental section, we will show these assumptions can degrade the performance of INFO-K-means. □

2.3 The Information-Theoretic View

Here, we derive the objective function in Equation (3) from an information-theoretic point of view. We begin our analysis by introducing an important lemma as follows.

Given a set of probabilistic distributions $\{p_1, p_2, \cdots, p_n\}$ and the corresponding weights $\{\pi_1, \pi_2, \cdots, \pi_n\}$, we have

LEMMA 1 (COVER&THOMAS,2006).

$$\sum_{i=1}^{n} \pi_i D(p_i || \sum_{i=1}^{n} \pi_i p_i) = H(\sum_{i=1}^{n} \pi_i p_i) - \sum_{i=1}^{n} \pi_i H(p_i). \quad (5)$$

Now, given a document data set $\mathbb{D}$, we want to partition $\mathbb{D}$ into K clusters without overlapping. Let random variables X, Y and C denote the document, term and cluster respectively, and x, y and c be the corresponding instances with $p(x)$, $p(y)$ and $p(c)$ as the occurrence probabilities. Furthermore, we assume that $p(c) = \sum_{x \in c} p(x)$. Then we have

THEOREM 2. *Let $I(X,Y)$ be the mutual information between discrete distributions X and Y, then*

$$I(X,Y) - I(C,Y) = \sum_{k} \sum_{x \in c_k} p(x) D(p(Y|x) \| p(Y|c_k)). \quad (6)$$

PROOF: By definition,

$$\begin{aligned} & I(X,Y) - I(C,Y) \\ &= \textstyle\sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} - \sum_k \sum_y p(c_k,y) \log \frac{p(c_k,y)}{p(c_k)p(y)} \\ &= \underbrace{\sum_x \sum_y p(y|x)p(x) \log p(y|x)}_{(a)} - \underbrace{\sum_k \sum_y p(y|c_k)p(c_k) \log p(y|c_k)}_{(b)} \\ &\quad \underbrace{- \sum_x \sum_y p(y|x)p(x) \log p(y)}_{(c)} + \underbrace{\sum_k \sum_y p(y|c_k)p(c_k) \log p(y)}_{(d)}. \end{aligned}$$

If we substitute $p(y|c_k) = \sum_{x \in c_k} \frac{p(x)}{p(c_k)} p(y|x)$ into (d), we have (c)=(d). Furthermore, it is easy to show

$$\begin{aligned} (a) &= -\textstyle\sum_k p(c_k) \left(\sum_{x \in c_k} \frac{p(x)}{p(c_k)} H(p(Y|x)) \right), \\ (b) &= -\textstyle\sum_k p(c_k) H \left(\sum_{x \in c_k} \frac{p(x)}{p(c_k)} p(Y|x) \right). \end{aligned}$$

By Lemma 1, we finally have Equation (6). □

REMARK: Theorem 2 illustrates the information theoretic view of INFO-K-means; that is, INFO-K-means tries to find a best partition on data so that the loss of mutual information due to the partitioning is minimized. □

In summary, we can have two different views on INFO-K-means. Both views provide the sound theoretic foundation. Furthermore, while Equation (6) is very similar to Equation (4), the information-theoretic view on INFO-K-means seems to be more appealing, since there is no prior assumption for $p(x)$ and $p(x|c)$, which is inevitable if we take the probabilistic view. In fact, we can view the probabilistic framework as a special case of the information-theoretic framework for INFO-K-means.

3. THE DILEMMA OF INFO-K-MEANS

In addition to having sound theoretic foundations, INFO-K-means has been widely shown that it has inferior performances to the spherical K-means on document clustering [21]. However, in this section, we illustrate an implementation challenge of INFO-K-means. We believe this challenge is one of the key issues that degrade the clustering performance of INFO-K-means.

For example, assume that we use INFO-K-means to cluster document data sets. To optimize the objective in Equation (3), we also launch the two-phase iteration process – reassigning instances to the "nearest" centroids, and updating the centroids according to the newly assigned instances subsequently. To this end, we must compute the KL-divergence values between each instance $p(Y|x)$ and each centroid $p(Y|c_k)$. In practice, we usually let

$$p(Y|x) = \frac{x}{n(x)},$$
$$p(Y|c_k) = \frac{p(x)}{\sum_{x \in c_k} p(x)} p(Y|x),$$

where $n(x)$ is the sum of all the term frequencies of x, $p(x)$ is the weight of x, as in Equations (4) and (6). Therefore, by Equation (2), to compute $D(p(Y|x) \| p(Y|c_k))$, we should expect that all the feature values of x are positive real numbers. Unfortunately, however, this is not the case for high-dimensional document data sets, which are famous for the sparseness in their high dimensions.

To illustrate this, we observe the computation of each dimension y. As we know,

$$D(p(Y|x) \| p(Y|c_k)) = \sum_{y} p(y|x) \log \frac{p(y|x)}{p(y|c_k)}.$$

To simplify the discussion, we denote $p(y|x) \log(p(y|x)/p(y|c_k))$ by D_y below. Then, the different combinations of $p(y|x)$ and $p(y|c_k)$ values can result in four different cases as follows.

1. *Case 1: $p(y|x) > 0$ and $p(y|c_k) > 0$.* In this case, the computation of D_y is straightforward, and the result can be any real number.

2. *Case 2: $p(y|x) = 0$ and $p(y|c_k) > 0$.* In this case, $\log(p(y|x)/p(y|c_k)) = \log 0 = -\infty$, which implies that the direct computation is infeasible. However, by the L' Hospital's rule [2], $\lim_{x \to 0^+} x \log(x/a) = 0$ $(a > 0)$. So we can let $x \equiv p(y|x)$, and thus have $D_y = 0$.

3. *Case 3: $p(y|x) > 0$ and $p(y|c_k) = 0$.* In this case, $D_y = +\infty$, which is hard to handle in practice.

4. *Case 4: $p(y|x) = 0$ and $p(y|c_k) = 0$.* In this case, we can simply omit this feature, or equivalently, we can let $D_y = 0$.

We summarize the above four cases in Table 1. As can be seen, for case 1 and 4, the computation of D_y is reasonable. However, the computation of D_y in case 2 has trouble; that is, it can not reveal any difference between $p(Y|x)$ and $p(Y|c_k)$ in dimension y, although $p(y|c_k)$ may be much larger than zero. Nevertheless, the most difficult case to handle is case 3. On the one hand, it is hard to do computations with $+\infty$ in practice. On the other hand, it is easy to know, if there is some dimension y of case 3, the total KL-divergence of $p(Y|x)$ and $p(Y|c_k)$ is infinite. This does not work for high dimensional sparse data sets, because the centroids of such data sets may typically contain many zero-value features. Therefore, we will have big challenges in assigning instances to the centroids, since the "instance-centroid" distances measured by the KL-divergence can be infinite. We call this problem as the zero-value dilemma.

Table 1: Cases in KL-divergence Computations.

	The Feature Value			
$p(y\|x)$	> 0	$= 0$	> 0	$= 0$
$p(y\|c_k)$	> 0	> 0	$= 0$	$= 0$
D_y	$\in \mathbb{R}$	0	$+\infty$	0

One way to solve the above dilemma is to smooth the sparse data sets. For instance, we can add a very small value to the entire data set so as to avoid having any zero feature value. While this smoothing technique indeed changes the sparseness property of the data sets, we will demonstrate in the experimental section that this method actually degrades the clustering performance of INFO-K-means.

In summary, there is a need to develop a new implementation scheme for INFO-K-means which should be able to avoid the zero-value dilemma.

4. THE SAIL SCHEME FOR INFO-K-MEANS

In this section, we propose a new implementation scheme, named SAIL, for INFO-K-means. To illustrate SAIL, we first present the concept of K-means distance, and then use K-means distance to simplify the objective function of INFO-K-means. Finally, we introduce the SAIL method.

4.1 The K-means Distance

Here, we briefly introduce the K-means distance. A detailed study of the K-means distance is available in [19].

DEFINITION 1 (K-MEANS DISTANCE). *We say that a distance function F is a K-means distance, if there exists some differentiable convex function $\phi : \mathbb{R}^d \to \mathbb{R}$ such that*

$$F(x, y) = \phi(x) - \phi(y) - (x - y)^t \nabla\phi(y).$$

In fact, any distance function that fits K-means must have the ability to facilitate the convergence of the two-phase iterations. So we have

LEMMA 2. *A distance function $F(x, y)$: $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ fits K-means[1], if and only if $\forall \mathbb{C} = \{x_1, x_2, \cdots, x_n\} \subset \mathbb{R}^d$,*

$$\overline{x} = \frac{\sum_{i=1}^n x_i}{n} \in \{y | \arg\min_{y \in \mathbb{R}^d} \sum_{i=1}^n F(x_i, y)\}$$

The K-means distance fits the K-means clustering. Indeed, under certain acceptable assumptions, the K-means distance is the only distance that fits K-means when the centroid type is the mean. That is,

THEOREM 3. *Assume that $F : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a non-negative function such that: (1) $F(x, x) = 0$, $\forall x \in \mathbb{R}^d$, (2) F and F_x are continuous, and (3) F_y is continuously differentiable on x. Then F fits K-means if and only if F is a K-means distance.*

The details and proofs can be found in [19]. We must point out that a K-means distance is not necessary to be a metric; that is, a K-means distance may not satisfy symmetry and triangle inequality properties. Also, the K-means distance is a family of distance functions with different ϕ, including the well-known Bregman divergence induced by strictly convex functions [1]. We list some popular K-means distances in Table 2.

We know that the KL-divergence belongs to the family of K-means distance. Specifically, according to Definition 1, KL-divergence can be rewritten as

$$D(x\|y) = -H(x) + H(y) + (x - y)^t \nabla H(y). \quad (7)$$

[1]Throughout this paper we focus on K-means with the arithmetic mean of cluster members as the centroid.

Table 2: Some K-means Distances.

$\phi(x)$	$F(x, y)$	Distance
$\|x\|^2$	$\|x - y\|^2$	Euclidian distance
$\|x\|$	$\|x\| - x^t y/\|y\|$	cosine distance
$-H(x)$	$D(x\|y)$	KL-divergence

4.2 The SAIL Method

In this subsection, we introduce the **S***ummation-B***A***sed* **I***ncremental* **L***earning* **(SAIL)** scheme for INFO-K-means. In general, SAIL has two distinct functions: 1) performing summations for each cluster during the clustering process; and 2) maintaining the summations incrementally.

To simply the discussion, we first transform the objective function in Equation (6) as follows.

THEOREM 4. *obj* : $\min \sum_k \sum_{x \in c_k} p(x) D(p(Y|x)\|p(Y|c_k))$ *is equivalent to*

$$obj: \ \min \sum_k p(c_k) H(p(Y|c_k)), \quad (8)$$

given $p(c_k) = \sum_{x \in c_k} p(x)$.

PROOF: By Formula (7), we have

$$D(p(Y|x)\|p(Y|c_k)) = -H(p(Y|x)) + H(p(Y|c_k))$$

$$+(p(Y|x) - p(Y|c_k))^t \nabla H(p(Y|c_k)).$$

Since

$$\sum_k \sum_{x \in c_k} p(x)(p(Y|x) - p(Y|c_k))^t \nabla H(p(Y|c_k)) = 0,$$

we have

$$\sum_k \sum_{x \in c_k} p(x) D(p(Y|x)\|p(Y|c_k))$$

$$= \underbrace{\sum_k p(c_k) H(p(Y|c_k))}_{(a)} - \underbrace{\sum_x p(x) H(p(Y|x))}_{(b)}.$$

It is easy to observe that (b) is constant for the given data set and the weights for the instances. Thus, the goal of minimizing the original objective function is equivalent to minimize (a). □

Next, based on the simplified objective function in (8), we establish the SAIL scheme for INFO-K-means. Generally speaking, SAIL is a greedy scheme which updates the objective function value "instance by instance". In other words, SAIL iteratively repeats the same procedure. First, SAIL randomly selects an instance from the data and assigns it to the most suitable cluster; then updates the objective function value and other related variables according to the assignment. This process is repeated until some stopping criterion is satisfied.

It is clear that SAIL differs from the traditional K-means. Indeed, SAIL is an incremental algorithm while the traditional K-means usually employs the batch learning mode.

Furthermore, SAIL also differs from the traditional incremental K-means; that is, to decide the assignment of each selected instance, SAIL does not compute the KL-divergence

values between the instance and all the centroids. Instead, it computes the Shannon entropies for the centroids which are assumed to be updated. This computation is supported by the two incrementally maintained summations for each cluster c: $p(c_k) = \sum_{x \in c} p(x)$ and $p(Y|c_k) = \sum_{x \in c} p(x)p(Y|x)$.

For instance, suppose SAIL randomly selects $p(Y|x')$ from a cluster $c_{k'}$. Then, if we assign $p(Y|x')$ to the cluster c_k, by the objective function in (8), the new objective value after this assignment will be as follows.

$$
\begin{aligned}
&obj(new) = obj(old) \\
&\underbrace{+ (p(c_{k'}) - p(x'))H\left(\frac{\sum_{x \in c_{k'}} p(x)p(Y|x) - p(x')p(Y|x')}{p(c_{k'}) - p(x')}\right)}_{(a)} \\
&\underbrace{- p(c_{k'})H\left(\frac{\sum_{x \in c_{k'}} p(x)p(Y|x)}{p(c_{k'})}\right)}_{(b)} \\
&\underbrace{+ (p(c_k) + p(x'))H\left(\frac{\sum_{x \in c_k} p(x)p(Y|x) + p(x')p(Y|x')}{p(c_k) + p(x')}\right)}_{(c)} \\
&\underbrace{- p(c_k)H\left(\frac{\sum_{x \in c_k} p(x)p(Y|x)}{p(c_k)}\right)}_{(d)},
\end{aligned}
$$

where (a)-(b) and (c)-(d) represent the two parts of changes on the objective function value due to the movement of $p(Y|x')$ from cluster $c_{k'}$ to cluster c_k. It is clear that the additivity of $\sum_{x \in c} p(x)$ and $\sum_{x \in c} p(x)p(Y|x)$ facilitates the computation. Then, among the new objective function values resulted by assigning the instance to all the clusters, we select the smallest one and assign the instance to the corresponding cluster. Finally, we update the two summations for $c_{k'}$ and c_k accordingly. This is the assigning routine of SAIL for each selected instance. Apparently, the two incrementally updating summations, i.e., $\sum_{x \in c} p(x)$ and $\sum_{x \in c} p(x)p(Y|x)$, are the keys for SAIL.

In summary, by the equivalent objective function in (8), the computations of the KL-divergences are replaced by the computations of Shannon entropies. As a result, we can avoid the zero-value dilemma.

4.3 A Description of the SAIL Algorithm

Figure 1 shows the pseudocode of the SAIL algorithm. Some implementation details are as follows.

Lines 1-3 are about data initialization. The preprocessing of $\mathbb{D}$ in line 2 includes the row and column modeling. This routine is also used to smooth the instances and/or assign weights to the features. Then, in line 3, we normalize the raw data instance x to $p(Y|x)$ where $p(y|x) = n(x, y)/n(x)$ for any feature y.

Lines 4-17 show the clustering process. Line 5 is for initialization, where $objVal$ denotes the objective function value, $label_{n \times 1}$ contains the cluster labels of the instances, $\pi_{K \times 1}$ stores the weight summation of the instances in each cluster, and $cluSum_{K \times d}$ stores the summation of the weighted instances in each cluster. That is, $\pi[k] = \sum_{x \in c_k} p(x)$ and $cluSum[k] = \sum_{x \in c_k} p(x)p(Y|c_k)$, for $k = 1, \cdots, K$, where n, d and K are the numbers of instances, features and clusters respectively. Two initialization modes have been employed in our implementation. In "Random Assignment" mode, we randomly assign the labels to the instances and then compute the required variables. But in "Random Read" mode, we read the whole instances one by one at random, and use a routine very similar to the clustering one to assign each instance. Lines 8-11 describe the clustering routine for each instance, which has been introduced in details in the previous subsection. Lines 13-14 show one stopping criterion in addition to the $maxIter$ parameter; that is, in one clustering if no instance changes its label after several iterations, we should stop this clustering. Finally, lines 18-19 choose the best clustering result among $reps$ clusterings.

SAIL (Summation Based Incremental Learning)

Input: $\mathbb{D}$: the data set.
π_x: the weight of $x \in \mathbb{D}$.
K: the number of clusters.
$reps$: the number of clusterings.
$maxIter$: the max number of iterations.
Output: $objVal^*$: the value of the obj. func. after clustering.
$label^*$: the cluster labels of the instances.

Procedure:

```
1.  Read 𝔻 and {π_x|x ∈ 𝔻};
2.  𝔻'=Preprocess(𝔻);
3.  ∀x ∈ 𝔻', x = Normalize(x);
4.  for i=1:reps
5.      Initialize(objVal(i), label(i), π(i), CluSum(i));
6.      for j=1:maxIters
7.          for l=1:n
8.              x = RandomSelect({x|x ∈ 𝔻'});
9.              ΔobjVal = TestAssign(x, π_x, π(i), CluSum(i));
10.             k = arg max_s{ΔobjVal[s], s = 1, ··· , K};
11.             Update(objVal(i), label(i), π(i), CluSum(i), k);
12.         end for
13.         if label(i) is unchanged in iteration l
14.             break;
15.         end if
16.     end for
17. end for
18. t = arg min_i objVal(i);
19. objVal* = objVal(t), label* = label(t);
```

Figure 1: The SAIL Algorithm.

Next, we briefly discuss the convergence issues and the computational complexity of SAIL. First, SAIL is a heuristic approach, so it does not guarantee to converge to a global minimum. While the objective function value decreases continuously (may not strictly) after reassigning each instance, and the combinations of the instances are limited, SAIL can still converge to some local minima. That is why we do multiple clusterings in SAIL and choose the best solution. In addition, SAIL also preserves the prominent advantage of the partitional clustering methods – low computational cost. Specially, the storage required is $O((n + K)d)$ and the time requirement for SAIL is $O(IKnd)$, where I is the number of iterations required for convergence, K is the number of clusters, d is the number of features, and n is the number of objects. Since K is small, I is often not beyond 20, and d can be substantially reduced due to the sparseness of data sets, the SAIL algorithm is extremely fast in our experiments.

5. EXPERIMENTAL RESULTS

In this section, we demonstrate the effectiveness of SAIL on improving the performance of INFO-K-means. Specifically, we show: 1) the traditional implementation of INFO-K-means with the KL-divergence is not effective on sparse data sets; 2) a simple smoothing technique is not a good so-

lution for the zero-value dilemma; and 3) SAIL based INFO-K-means can achieve superior performances to the smoothing method as well as the spherical K-means.

5.1 The Experimental Setup

Experimental Tools. In the experiments, we employ four types of clustering tools. The first one is our developed SAIL-based INFO-K-means. The other three are well-known software packages for K-means clustering, including MATLAB v7.1 [14], CO-CLUSTER v1.1 [5], and CLUTO v2.1.1 [11].

The MATLAB implementation of K-means is the batch-learning version which must compute the distances between instances and centroids. We extend it to include more distance functions, such as KL-divergence. Therefore, it is a centroid-based implementation of INFO-K-means which has to compute the KL-divergence directly.

CO-CLUSTER is a C++ program which implements the information theoretic co-clustering algorithm [7]. Although it still computes the "instance-centroid" KL-divergences, it provides various methods to improve the clustering performances such as annealing, batch and local search, etc.

CLUTO is a software package for clustering high dimensional data sets. Specifically, its K-means implementation with cosine similarity as the proximity shows superior performances on clustering document data sets and gene expression data sets [20]. In the experiments, we compare CLUTO with our SAIL-based INFO-K-means on a number of real-world data sets.

Note that the parameters of the four K-means implementations were set to match one another for the purpose of the comparison, and the cluster number K was set to match the class number of each data set.

Validation Measures. Many recent studies on clustering used the Normalized Mutual Information (NMI) to evaluate the clustering performance [21]. For consistency, we also use NMI in our experiments, which can be computed as: $NMI = I(X,Y)/\sqrt{H(X)H(Y)}$, where the random variables X and Y denote the cluster and class sizes, respectively. NMI values are in [0,1], and a larger NMI value indicates a better clustering result.

Moreover, we use Coefficient of Variation (CV) [4] to measure the dispersion degree of the cluster sizes. The CV is defined as the ratio of the standard deviation to the mean. Given a set of objects $X = \{x_1, x_2, \ldots, x_n\}$, we have $CV = s/\bar{x}$ where $\bar{x} = \sum_{i=1}^{n} x_i/n$ and $s = \sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2/(n-1)}$. CV is a dimensionless number that allows comparison of the variation of populations that have significantly different mean values. In general, the larger the CV value is, the greater the variability in the data.

Experimental Data Sets. For our experiments, we used a number of real-world document data sets. Some characteristics of these data sets are shown in Table 3.

The `fbis` data set was from the Foreign Broadcast Information Service data of the TREC-5 collection [18], and the classes correspond to the categorization used in that collection. The `sports` data set was derived from the San Jose Mercury newspaper articles that were distributed as part of the TREC collection (TIPSTER Vol. 3). It contains documents about baseball, basketball, bicycling, boxing, football, golfing, and hockey. Data sets `tr11`, `tr12`, `tr23`, `tr31`, `tr41` and `tr45` were derived from TREC-5 [18], TREC-6 [18], and TREC-7 [18] collections. The classes of these data sets correspond to the documents that were judged relevant to particular queries. Data sets `la1`, `la2` and `la12` were obtained from articles of Los Angeles Times that was used in TREC-5 [18]. The categories include documents from the entertainment, financial, foreign, metro, national, and sports desks. The `ohscal` data set was obtained from the OHSUMED collection [10], which contains documents from various biological sub-fields. Data sets `k1a`, `k1b` and `wap` were from the WebACE project [9]; each document corresponds to a web page listed in the subject hierarchy of Yahoo!. In particular, `k1a` and `k1b` contain exactly the same set of documents but the former contains a finer-grain categorization. The `classic` data set was obtained by combining the CACM, CISI, CRANFIELD, and MEDLINE abstracts that were used in the past to evaluate various information retrieval systems. Data set `cranmed` was attained similarly. Finally, the data set `re0` was from Reuters-21578 collection Distribution 1.0 [12]. For all data sets, we used a stop-list to remove common words, and the words were stemmed using Porter's suffix-stripping algorithm [15].

Table 3: Experimental Data Sets.

Dataset	Source	#Case	#Attr.	#Class	CV_0
fbis	FBIS (TREC)	2463	2000	17	0.961
sports	San Jose Mercury (TREC)	8580	126373	7	1.022
tr11	TREC	414	6429	9	0.882
tr12	TREC	313	5804	8	0.638
tr23	TREC	204	5832	6	0.935
tr31	TREC	927	10128	7	0.936
tr41	TREC	878	7454	10	0.913
tr45	TREC	690	8261	10	0.669
la1	LA Times (TREC)	3204	21604	6	0.493
la2	LA Times (TREC)	3075	31472	6	0.516
la12	LA Times (TREC)	6279	31472	6	0.503
ohscal	OHSUMED-233445	11162	11465	10	0.266
k1a	WebACE	2340	21839	20	1.004
k1b	WebACE	2340	21839	6	1.316
wap	WebACE	1560	8460	20	1.040
classic	CACM/CISI/CRAN/MED	7094	41681	4	0.547
cranmed	CRAN/MED	2431	41681	2	0.212
re0	Reuters-21578	1504	2886	13	1.502

5.2 The Effect of the Zero-Value Dilemma

Here, we demonstrate the effect of the zero-value dilemma. Specifically, we show the difficulties of the traditional implementation of INFO-K-means with the KL-divergence on sparse data sets. Since MATLAB K-means can handle infinity (denoted by INF), we selected `tr23` and `tr45` as the test data sets and applied MATLAB K-means for testing without smoothing. The clustering results are shown in Table 4, in which CV_0 and CV_1 represent the distributions of the class sizes and cluster sizes, respectively.

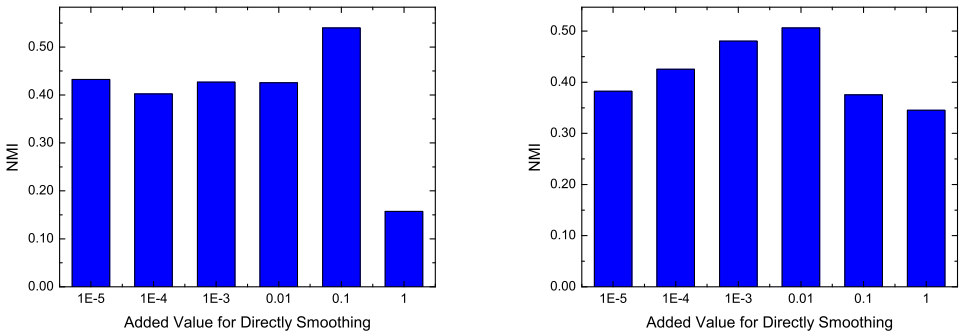
As indicated by the close-to-zero NMI values, the clustering performance of MATLAB K-means without smoothing is extremely poor. Also, by comparing the CV_0 and CV_1 values, we found that the distributions of the resultant clus-

Table 4: Clustering Results of MATLAB K-means.

Data Set	NMI	CV_0	CV_1
`tr23`	0.035	0.935	2.435
`tr45`	0.022	0.669	3.157

Table 5: Clustering Results of CO-CLUSTER.

Data Set	Search	Annealing	NMI	CV_0	CV_1
tr12	batch	1.0	0.058	0.638	0.295
		0.5	0.040	0.638	0.374
		none	0.031	0.638	0.376
	local	1.0	0.045	0.638	0.334
		0.5	0.059	0.638	0.339
		none	0.048	0.638	0.461
tr31	batch	1.0	0.007	0.936	0.426
		0.5	0.011	0.936	0.362
		none	0.010	0.936	0.405
	local	1.0	0.014	0.936	0.448
		0.5	0.009	0.936	0.354
		none	0.011	0.936	0.365



(a) `tr11` Data Set (b) `tr45` Data Set

Figure 2: The Effect of Data Smoothing.

ter sizes are much more skewed than the distributions of the class sizes. In fact, for both test data sets, nearly all the data instances have been assigned to ONE cluster! Therefore, this experiment result confirms our analysis in Section 3 on the zero-value dilemma on sparse document data sets.

Furthermore, we tested CO-CLUSTER on `tr12` and `tr31` data sets. As mentioned above, CO-CLUSTER also computes the KL-divergence values in the clustering process, but it provides different search modes and the annealing technique to avoid poor local minima. Table 5 shows the clustering results, where "Search" and "Annealing" indicate the search modes and annealing parameters respectively.

In Table 5, we can observe that the use of annealing technique and different search modes does not improve the clustering performance. The near-to-zero NMI values indicate the poor clustering performance. This again confirms that the direct computation of the KL-divergence values is infeasible for sparse data sets. Another interesting observation is that, the clusters produced by CO-CLUSTER are much more balanced than the clusters produced by MATLAB K-means as indicated by the much smaller CV_1 values.

5.3 SAIL versus Smoothing

Here, we first illustrate the effect of the smoothing technique on sparse data sets. In this experiment, we use MATLAB K-means and select seven document data sets. Figure 2(a) and 2(b) show the clustering results on data sets `tr11` and `tr45`, where the added small values gradually increase along the horizon axis.

One observation is that data smoothing indeed improves the clustering performance of INFO-K-means. This further justifies our observation on the zero-value dilemma for INFO-K-means with the KL-divergence as the proximity function. Another interesting observation is that the optimal added values (OAV) are different for different data sets. For instance, while $OAV \approx 0.1$ for `tr11`, $OAV \approx 0.01$ for `tr45`. This implies one issue with data smoothing in practice; that is, it is difficult to have the optimal smoothing effect. Nevertheless, we should avoid setting extreme values for added values. A tiny added value may not mitigate the zero-value dilemma, but a large value may damage the integrity of the data instances, and thus leads to the degraded clustering performance. Figure 2(b) well illustrates this.

For the purpose of comparison, we also tested SAIL on these data sets. The parameters were set by default as follows. 1) $\pi_x = 1/n$, for any $x \in \mathbb{D}$; 2) no row or column modeling in step 2 of Figure 1; 3) the initialization mode of the variables in step 5 of Figure 1 is "random read".

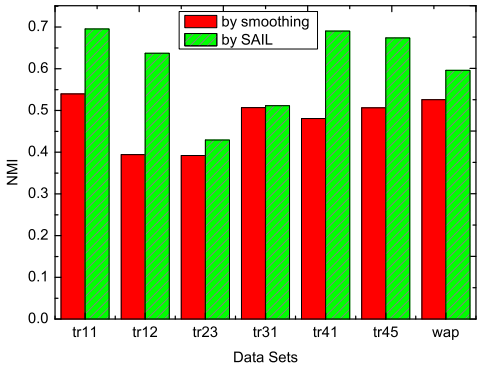


Figure 3: Smoothing vs. SAIL.

Figure 3 shows the comparison results. As can be seen, SAIL shows consistently superior performances to the smoothing technique. For some data sets such as `tr11`, `tr12`, `tr41` and `tr45`, SAIL takes the lead with a wide margin. Please note that, for the smoothing method, we tried a series of added numbers, i.e., 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, and selected the best one for comparison.

In summary, while the traditional data smoothing technique can improve the performance of INFO-K-means on sparse data sets, it changes the data integrity and has issues on setting the optimal data value added into the data. In contrast, SAIL has no parameter setting issue and can lead to consistent better clustering performances than the smoothing technique. This indicates that SAIL is a better solution for the zero-value dilemma.

5.4 SAIL versus Spherical K-means

In this subsection, we compare the clustering performance between SAIL and the spherical K-means. In the literature, people have shown that spherical K-means usually produces better clustering results than traditional K-means [21]. And the CLUTO version of the spherical K-means [11] even shows superior performances on document data sets, which makes it to be the benchmark method for document clustering. However, we would like to show in the experiment that the performance of SAIL is comparable to or even slightly better than the spherical K-means in CLUTO.

Table 6: Spherical K-means vs. SAIL.

Data Set	CLUTO		SAIL	Density
	none-IDF	IDF		
classic	0.450	0.561	0.678	0.0008
cranmed	0.900	0.976	0.990	0.0014
fbis	0.579	0.603	0.612	0.0799
k1a	0.574	0.549	0.584	0.0068
k1b	0.589	0.583	0.623	0.0068
la1	0.360	0.571	0.585	0.0048
la12	0.396	0.000	0.586	0.0048
la2	0.355	0.443	0.537	0.0047
ohscal	0.347	0.606	0.438	0.0053
re0	0.411	0.666	0.430	0.0179
sports	0.442	0.583	0.638	0.0010
tr11	0.628	0.111	0.696	0.0438
tr12	0.634	0.401	0.637	0.0471
tr23	0.271	0.493	0.429	0.0661
tr31	0.411	0.543	0.511	0.0265
tr41	0.551	0.244	0.690	0.0262
tr45	0.554	0.364	0.674	0.0340
wap	0.567	0.679	0.596	0.0167

Note: Measured by NMI.

In this experiment, the parameter settings in CLUTO are as follows: clmethod=direct, crfun=i2, sim=cosine, ntrials=10. Since the column modeling makes great impact on the spherical K-means, we provide clusterings with and without column modeling, corresponding to "IDF" (Inverse Document Frequency) and "none-IDF" in Table 6. For SAIL, we used the default settings as the previous experiments. Table 6 shows the clustering results.

As can be seen, if matched parameters are used (i.e., "none-IDF" for the spherical K-means), SAIL shows consistent better clustering quality on all 18 data sets. This can be observed in Figure 4: the NMI curve of the spherical K-means is completely included in the NMI curve of SAIL.

The Spherical K-means with IDF shows more appealing clustering results. As can be seen in Table 6, the clustering performances of 11 out of 18 data sets have been improved with IDF. In particular, for data sets `ohscal`, `re0`, `tr23`, `tr31` and `wap`, the clustering results are even better than that of SAIL. Nevertheless, we should also notice that the spherical K-means with IDF also degrades the clustering performances of the rest 7 data sets. For data sets `la12`, `tr11`, `tr12`, `tr41` and `tr45`, the negative impact is quite substantial. This implies that column modeling is an X factor for the performance of the spherical K-means. That is, for data sets without class information, whether to use column modeling in the spherical K-means is a dilemma: it may improve or degrade the clustering performance. In contrast, SAIL with default settings shows consistent clustering performances and is more robust in practice.

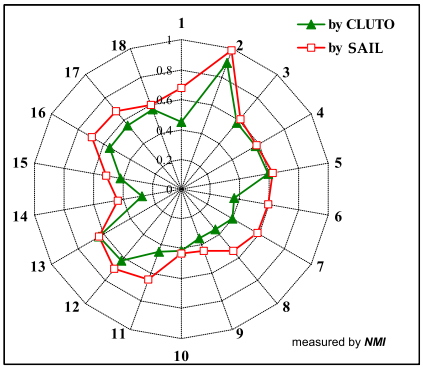


Figure 4: Spherical K-means vs. SAIL.

Table 7: Impact of Parameter Settings on SAIL.

Data Set	Default	Column Modeling	Row Weighing	
		(IDF)	(Entropy)	(Sum)
tr11	0.696	0.614	0.669	0.644
tr12	0.637	0.501	0.574	0.474
tr23	0.429	0.402	0.368	0.171
tr31	0.511	0.491	0.428	**0.540**
tr41	0.690	0.618	0.669	**0.756**
tr45	0.674	**0.750**	0.653	0.563
wap	0.596	0.573	**0.606**	0.593

Note: Measured by NMI.

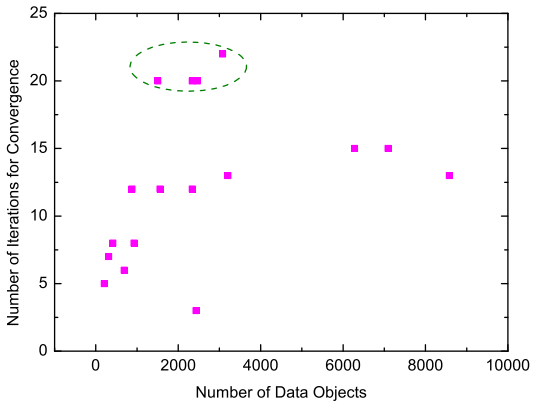


Figure 5: The Convergent Speed of SAIL.

5.5 Robustness and Computation Performance

In this subsection, we investigate the robustness and computation performance of SAIL.

First, we observe the impact of data sparseness on SAIL. We compute the "density", i.e., the number of none-zero values to the number of all values, for each document data set in Table 6. Thus we have the correlation between "SAIL" and "Density" columns: -0.16. This implies that the performance of SAIL has no strong correlation with data sparseness. In other words, the performance of SAIL is not sensitive to the data sparseness. This means that SAIL is a good solution to the zero-value dilemma.

Second, we observe the impact of parameter settings on SAIL. Table 7 shows the results, where "Default" means no row or column modeling and the weights are the same, "Column Modeling" employs IDF only, and "Row Weighing" uses entropy-based and sum-based weighing methods respectively. Note that, for each instance x, the entropy-based method uses $1/H(p(Y|x))$ as the weight, while the summation-based method uses $n(x)$ as the weight.

As can be seen in the table, column modeling and row weighing improve the clustering performances of SAIL on several data sets, as highlighted by bold font. However, in many cases, SAIL with default settings can produce satisfying results already. Therefore, SAIL is relatively robust with the parameter settings.

Finally, Figure 5 shows the relationship between the number of data instances and the number of iterations for convergence. In the figure, we can observe that the number of iterations for convergence (NIC) of SAIL is no more than 20 except for `la2` and `ohscal`. Also, NIC increases slowly as the rapid increase of the number of objects n.

6. RELATED WORK

In the literature, great research efforts have been taken to incorporate information theoretic measures into exist-

ing clustering algorithms, such as K-means. However, the zero-denominator dilemma remains a critical challenge. For instance, Dhillon et al. proposed information-theoretic K-means, which used the KL-divergence as the proximity function [6]. While the authors noticed the "infinity" values when computing the KL-divergence, they did not provide specific solutions to this dilemma. In addition, Dhillon et al further extended information-theoretic K-means to the so-called information-theoretic co-clustering [7]. This algorithm is the two-dimensional version of information theoretic K-means which monotonically increases the preserved mutual information by interwinding both the row and column clusterings at all stages. Also, there is no solution provided for handling the zero-value dilemma when computing the KL-divergence. Finally, the information bottleneck (IB) is similar to INFO-K-means in preserving mutual information [17]. In [16], Slonim and Tishby also found that the IB-based word clustering can lead to the zero-value dilemma. They suggested to use the smoothing method by adding 0.5 to each entry of the document data set.

This paper indeed fills this crucial void. Specifically, in our SAIL method, the computation of the KL-divergence is replaced by the computation of the Shannon entropy. This helps to avoid the zero-value dilemma.

7. CONCLUDING REMARKS

This paper studied the problem of exploiting information theoretic measures, such as the KL divergence and the Shannon Entropy, as the clustering criteria for K-means clustering. In particular, we revealed the dilemma of the KL divergence for handling high-dimensional sparse data; that is, the centroids in sparse data usually contain zero-value features, and thus lead to infinite KL divergence values. This makes it difficult to use the KL divergence as a criterion for assigning objects to the centroids. To that end, we developed a Summation-based Incremental Learning (SAIL) method, which can avoid the zero-value dilemma by using the Shannon entropy instead of the KL divergence. This replacement is guaranteed by an equivalent mathematical transformation in the K-means objective function. Finally, as demonstrated in our experiments, SAIL can greatly improve the performance of K-means on high-dimensional sparse data.

8. ACKNOWLEDGEMENTS

This research was partially supported by the National Science Foundation of China (NSFC) (No. 70621061, 70321010), and the Rutgers Seed Funding for Collaborative Computing Research. Also, this research was supported in part by a Faculty Research Grant from Rutgers Business School- Newark and New Brunswick.

9. REFERENCES

[1] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

[2] L. Brand. *Advanced Calculus: An Introduction to Classical Analysis*. Dover Publications, 2006.

[3] T. Cover and J. Thomas. *Elements of Information Theory (2nd Edition)*. Wiley-Interscience, 2006.

[4] M. DeGroot and M. Schervish. *Probability and Statistics (3rd Edition)*. Addison Wesley, 2001.

[5] I. Dhillon. Co-clustering software, version 1.1. http://www.cs.utexas.edu/ inderjit/software.shtml.

[6] I. Dhillon, S. Mallela, and R. Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, 3:1265–1287, 2003.

[7] I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 89–98, 2003.

[8] C. Elkan. Clustering documents with an exponential-family approximation of the dirichlet compound multinomial distribution. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 289–296, 2006.

[9] E.-H. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Webace: A web agent for document categorization and exploration. In *Proceedings of the 2nd International Conference on Autonomous Agents*, pages 408–415, 1998.

[10] W. Hersh, C. Buckley, T. Leone, and D. Hickam. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 192–201, 1994.

[11] G. Karypis. Cluto-version 2.1.1. http://glaros.dtc.umn.edu/gkhome/views/cluto.

[12] D. Lewis. Reuters-21578. http://www.daviddlewis.com/resources/testcollections/reuters21578/.

[13] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[14] MathWorks. K-means clustering in statistics toolbox. http://www.mathworks.com.

[15] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[16] N. Slonim and N. Tishby. The power of word clusters for text classification. In *Proceedings of the 23rd European Colloquium on Information Retrieval Research*, 2001.

[17] N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing*, 1999.

[18] TREC. Text retrieval conference. http://trec.nist.gov.

[19] J. Wu, H. Xiong, J. Chen, and W. Zhou. A generalization of proximity functions for k-means. In *Proceedings of the 2007 IEEE International Conference on Data Mining*, pages 361–370, 2007.

[20] Y. Zhao and G. Karypis. Criterion functions for document clustering: Experiments and analysis. *Machine Learning*, 55(3):311–331, 2004.

[21] S. Zhong and J. Ghosh. Generative model-based document clustering: A comparative study. *Knowledge and Information Systems*, 8(3):374–384, 2005.