A Scalable Algorithm for Metric High-Quality Clustering in Information Retrieval Tasks

Filippo Geraci
Istituto di Informatica e Telematica,
Consiglio Nazionale delle Ricerche, Pisa
and
Dipartimento di Ingegneria dell'informazione
Università di Siena
Italy
Email: filippo.geraci@iit.cnr.it

Marco Pellegrini
Istituto di Informatica e Telematica,
Consiglio Nazionale delle Ricerche
Pisa, Italy
Email: marco.pellegrini@iit.cnr.it

Paolo Pisati
Istituto di Informatica e Telematica,
Consiglio Nazionale delle Ricerche, Pisa
and
Università Statale di Milano
Italy
Email: paolo.pisati@iit.cnr.it

***Abstract*— We consider the problem of finding efficiently a high quality k-clustering of n points in a (possibly discrete) metric space. Many methods are known when the point are vectors in a real vector space, and the distance function is a standard geometric distance such as L_1, L_2 (Euclidean) or L_2^2 (squared Euclidean distance). In such cases efficiency is often sought via sophisticated multidimensional search structures for speeding up nearest neighbor queries (e.g. variants of kd-trees). Such techniques usually work well in spaces of moderately high dimension (say up to 6 or 8). Our target is a scenario in which either the metric space cannot be mapped into a vector space, or, if this mapping is possible, the dimension of such a space is so high as to rule out the use of the above mentioned techniques. This setting is rather typical in Information Retrieval applications. We augment the well known *furthest-point-first* algorithm for *k-center* clustering in metric spaces with a filtering step based on the triangular inequality and we compare this algorithm with some recent fast variants of the classical *k-means* iterative algorithm augmented with an analogous filtering schemes. We extensively tested the two solutions on synthetic geometric data and real data from Information Retrieval applications. The main conclusion we draw is that our modified furthest-point-first method attains solutions of better or comparable quality within a fraction of the time used by the fast k-means algorithm. Thus our algorithm is valuable when either real time constraints or the large amount of data highlight the poor scalability of traditional clustering methods.**

I. Introduction

"Clustering" is an important operation in the exploratory analysis of large data sets. Intuitively, given a data set from some domain, data "points" that are "similar" with each other should belong to the same cluster, as a consequence the partition of the data set into clusters makes explicit the structure of the "data space" so to facilitate further human or automatic analysis.

Clustering has a wide range of applications in areas such as data mining, text mining, pattern recognition, quantization, and expression of genomic data. Classical textbooks (see e.g. [JD88], [Har75], [DO74], [And73] and [Zup82]) and surveys (see e.g. [JMF99] and [Ber02]) contain hundreds of bibliographic references on the issue of clustering.

Tools for clustering results of several web search engines have recently become a focus of attention in the research community due to the commercial success of services provided, for example, by *Vivisimo*, *Dogpile*, and *Kartoo* (see also [GRS04]). Real-time high quality clustering is a key ingredient of such systems[1].

Overall one can distinguish several general purpose algorithms of wide applicability, as well as many domain-specific methods developed within each discipline. In this paper we consider mostly general purpose algorithms and the experiments described are meant to give examples of the effectiveness of the method mainly on data sets found in Information Retrieval tasks.

More generally in this paper we consider the following type of applicative scenario:

a) the input data "points" admit a distance function that is a metric, in particular, the triangular inequality is satisfied;
b) computing the distance function for a pair of "points" is expensive, thus a primary goal is to minimize the number of distance evaluations;
c) the requirement of finding an high quality clustering is balanced by the need to find such high quality clustering quickly;
d) the number of data points n and the target number of clusters of clusters k are both large, where "large" here must be considered in the context of the required response time;
e) when the data set can be mapped in a real vector space R^d, the dimension d of such a space is also large.

As in many other areas of algorithmic research empirical performance and provable complexity/quality properties are in general difficult to pursue together. Recently, using techniques from random sampling and approximation theory, new algorithm have been produced with good performance/quality

[1]However a second essential ingredient, not treated here, is that of finding significant labelling of the clusters found to convey useful information to the user.

Work partially supported by the Italian Registry of ccTLD .it.

guarantees in Euclidean space (e.g. [AP98] and [Mat00]) or metric spaces [Ind99], whose value in an empirical setting has yet to be established. On the other hand an even more difficult task is to analyze formally the performance of well established clustering algorithm such as the k-means algorithm [HPS05]. Our course is an intermediate one: we start with an algorithm that has performance guarantees and we augment it with heuristics, based on the triangular inequality and random sampling, so to attain good empirical performance, which is demonstrated via carefully designed experiments.

Our algorithm is a variation of the furthest-point-heuristic for the k-center problem [Gon85], [HS85], [DF85]. Taking as measure of quality the minimum of the maximum diameter of a cluster over all possible k-clustering of set of n points in a metric space, the solution found by the furthest-point-heuristic is within a factor 2 of the optimum. This approximation factor is tight for polynomial algorithms in metric spaces and almost tight in Euclidean spaces, unless P=NP. A straightforward implementation of the furthest-point-heuristic would cost $O(nk)$ distance evaluations. In this paper we modify the furthest-point-heuristic by using the triangular inequality to filter out useless distance computation, thus significantly speeding up the computation in practice with respect to the straightforward implementation. Moreover we will show that the speed/quality performances of our modified furthest-point-heuristic is competitive with those attained by recently modified versions of k-means described by Phillips [Phi02] that also exploit filters based on the triangular inequality.

Clustering is such a very important but also rather vague task, so to give rise to a very large spectrum of tools, models and approaches. Here we restrict ourselves to consider algorithms for which some provable guarantee in time, quality or both, has been established and those that for their wide use are considered the standard empirical benchmark, namely Lloyd's algorithm (aka. K-means algorithm), including some recent improved variants. In the context of on-line applications in Information Retrieval, the scatter/gather algorithm described in [CPKT92], [CKP93], [HP96] is often used as a benchmark algorithm. Scatter/gather is a variant of k-means with additional split/join operations on clusters intended to improve the quality of those clusters of lower quality. The initialization and the first refinement steps have complexity $O(nk)$ and there in no speed up due to the use of the triangular inequalities, therefore scatter/gather incurs in the high time costs of the standard k-means algorithm. For this reason we concluded that Phillips' variant [Phi02] is a fairer benchmark for our tests.

In Section V the experimental setting and results are described in detail. Here we give a brief summary. Experiments with the synthetic data set BIRCH, with 100,000 points and 100 classes in 2D, which is considered challenging for clustering algorithms, are shown in table I. In this experiment k-center is faster than all of the initializations of k-means we tested, and has better quality. After ten iterations k-mans attains quality slightly superior to k-center, but with a higher time cost by two orders of magnitude. Experiments with the Reuters collection of 8528 documents and 55 classes are shown in Table II. In this experiment k-center achieves quality levels comparable to those attained by k-means after ten iterations, but uses substantially less time. Among the initializations of k-means only one based on sampling is as fast as k-center but yields a lower quality in F-measure. In experiments with collections of snippets generated on-line by querying a large directory of web pages (ODP) results of comparable and often better quality are attained by k-center while the gain in time is of a factor ranging from 5 to 10. Although more experiments are surely needed to fine tune the technique these findings give us a fair confidence on the validity of the proposed algorithm

The paper is organized as follows. We introduce formalism and notation in Section II. The main algorithms are described in Section III and our heuristic modifications in Section IV. The experimental setting and the results are described in Section V. A survey of the main results on clustering is in Sections VI and VII.

II. Formalism and notation

A. Internal and external quality criteria

The quality of the output of a clustering algorithm can be evaluated in several ways, but a basic distinction can be made between *internal end external criteria*. An *external criterion* is one in which the outcome of the algorithm is compared against the hidden classification of the input points that the algorithm is supposed to find out, also called the *ground truth*. In an unsupervised learning setting the algorithm has no (or very little) a priori information on the structure of the ground truth, and only the general soundness of the overall data model in association with a well designed algorithm can lead to a clustering of high quality.

On the other hand an *internal criterion* is one in which the outcome of the algorithm is compared against a functional that is in principle computable (for example via exhaustive enumeration) from the same input data that are fed to the clustering algorithm. In general, the form of the functional is one of the ingredients that are used on the design of the clustering algorithm. When inner criteria are used the initial data modelling phase has little impact and the main item being gouged is the clustering algorithm itself.

Formal proofs of cluster quality can be made only with respect to internal criteria, since these are formal objects, while external criteria are not formalized (and often, indeed, not formalizable). While it is fair to use internal criteria to compare different algorithms provided these are designed for the same internal criterion, no such restriction applies if the external criterion is used. Besides external criteria are closer to the notion of "user satisfaction" which is rather informal but very important in Information Retrieval applications.

Since we compare two algorithms developed for different internal criteria, we shall use in our tests an external criterion comparison. In the next section we formalize these concepts.

B. Formalization of the problem

We have a finite or infinite class of objects $\mathcal{C}$ and finite subset $S \subseteq \mathcal{C}$, where we denote with n the cardinality of S. We stipulate the existence of a labelling function $f_S : S \to L$ associating to each element of S a label form a denumerable set L of labels. To simplify[2] our setting we identify L with the set of natural numbers N. Note that generally f depends on S. On the other hand, when f does not depend on S, it is just the restriction to S of a function $F : \mathcal{C} \to L$. This would be a restricted and simpler case.[3]

Given S, determine f_S is our goal, or, more precisely, determine the partition of S into equivalence classes induced by f^{-1}; we denote this partition with $P(S, F_S)$:

$$P(S, f_S) = \{A \in 2^S | \exists l \in L \ : \ A = f_S^{-1}(l)\}.$$

Note that this second formulation relieves us from the burden of determining exactly the nature of the label set L or from the temptation of using special properties of clever labelling schemes.

What is known, provided a computational cost is paid in its point-wise evaluation, is a distance function $D : S \times S \to R_{\geq 0}$ associating a non negative real number to a pair of elements of S. A proper modelling of the problem aims at the devising a distance D that, for every S, is related to the partition $P(S, f_S)$ and is computable.

It is a natural step forward to interpret D as a metric (thus imposing or assuming) that satisfies the three axioms (for every $s, p \in S$, $D(s, s) = 0$, $D(s, p) = D(p, s)$ and the triangular inequality (metric case).

A more elaborated model requires a mapping of the elements of S to points in a Real Vector Space V, $m_V : S \to V$, while the distance D is a normed distance in such space. The use of vector space model introduces a large quantity of possibilities, in particular it is possible to generate new points in a vector space as a linear combination of other points. Note that points so constructed are not necessarily in relations to elements of S (or of $\mathcal{C}$), thus properties of D defined for points in $m_V(S)$ might not hold for other points in V.

In many applications the Vector space model is an obvious one or at least a simple one to conceive. For example protein expression data in μ-arrays are naturally modelled as a two-dimensional matrix, thus suggesting to consider the columns as dimensions of the real vector space and the rows as point in such space (or viceversa). For information retrieval applications mapping a text as point in an high dimensional vector space using the TF-IDF model is by now a standard tool. Is some other areas such as clustering of proteins based on similarity of their DNA encoding strings, instead the mapping to a vector space is more problematic, since there is little biological significance to be attached to a linear combination of encoding strings. In this case a metric model is more natural.

One feature to be considered in the choice of the clustering algorithm is the role of transitivity. That is, if a is close to b and b is close to c is this sufficient to conclude that a, b, c might be in the same cluster, or it must also hold a close to c? In a 2-dimensional example with points, the first choice allows the formation of arbitrary shaped clusters, while the second choice forces the creation of ball-like clusters. The first choice might lead to discovering new and unexpected associations, but is also less robust to noise and spurious chaining effects.

Since determining automatically the correct number of clusters is a difficult problem in itself, we simplify the setting of the comparison by passing to our algorithm as a parameter the number of clusters in the ground truth.

C. Problems: k-center, k-medians, k-means

The following are some classical internal functionals to be minimized.

1) The k-center problem: Given S a point set in vector space V, endowed with a distance function D, and k, partition S into subsets $C_1, ..C_k$ and determine points $\mu_1, ..\mu_k \in V$ so that the maximum cluster radius is minimized :

$$\min \max_j \max_{x \in C_j} D(x, \mu_j)$$

A second equivalent definition is as follows. For a point p and a set of points X, we set $D(p, X) = \min_{q \in X} D(p, q)$, thus extending the notion if distance from point-to-point to point-to-set. Given the input set S and a set of centers X the partition of S is implicit. We can define the k-center problem as the problem of finding a set X of k points so to find:

$$\min_X \max_{p \in S} D(p, X)$$

2) The k-medians problem: Given S and k, partition S into subsets $C_1, ..C_k$ and determine points $\mu_1, ..\mu_k \in V$ so that the sum of all the point-center distances is minimized:

$$\min \sum_j \sum_{x \in C_j} D(x, \mu_j).$$

In the more compact notation this becomes the problem of finding:

$$\min_X \sum_{p \in S} D(p, X).$$

The 1-median problem (finding a point minimizing the sum of distances from all other points in the set) is also known as the Fermat-Weber problem and does not have a closed formula solution.

3) The k-means problem: Given S and k, partition S into subsets $C_1, ..C_k$ and determine points $\mu_1, ..\mu_k \in V$ so that so that sum of squares of inter-cluster point-center distances is minimized:

$$\min \sum_j \sum_{x \in C_j} (D(x, \mu_j))^2.$$

[2] In some application areas, such as clustering of results from web searching engines the labelling task is not a trivial one.

[3] In the second case we have a universal labelling, in the first a local one. In certain application the local model might be more appropriate, in some the universal model is more appropriate.

In the more compact notation this becomes the problem of finding:

$$\min_X \sum_{p \in S} (D(p, X))^2$$

The value of this functional for a given clustering is also called the *squared error distortion* of the clustering.

4) The generalized k-means problem: Sometimes a more general objective function is used, defined as:

$$\min \sum_j |C_j|^\alpha \sum_{x \in C_j} (D(x, \mu_j))^2$$

where for $\alpha = 0$ one gets the previous formula, for $\alpha = 1$ one gets the sum of average point center distances. This more general formulation is used for example in [IKI94]. Since in the continuous case it can be shown that $\mu_j = (1/|C_j|) \sum_{p \in C_j} p$ that is the optimal center of a cluster is the centroid of the cluster, simple algebraic manipulations show that for $\alpha = 2$ one gets the sum of squares of all pairwise distances of points in clusters.

D. Variants

If we impose that also $\mu_1, ..\mu_k \in S$ we have a "combinatorial" version of the above problems. A continuous version of the above problem is one in which the set of points is not discrete (while the set of centers is discrete) [FMW00].

E. Graph based measures

The classical k-center, k-median and k-mean formulations for the clustering problem are very popular but by no means the only one. Recently in [DFK+99], [KVV00], [CKVW03], [Dhi01], clustering criteria based on the notion of the conductance of a cut in a graph have been proposed together with approximation algorithms based on Singular Value Decomposition computations. Other classical graph-based clustering are obtained via the construction of Minimum (and maximum) Spanning Trees of the distance graph of the input points [ABKY88].

III. Algorithms

A. The furthest-point-first method

In [Gon85] it is shown an algorithm called APPROX in that paper that finds a solution within a constant factor of the optimum for the k-center problem when only the metric space model is assumed. Basically the same solution called furthest-point-traversal has been found also by Hochbaum and Shmoys [HS85], although in the context of clustering of a graph.

In [FG88] the same algorithm is christened "Furthest point algorithm" and described in a slightly different language. Given the set S of points, and a set $T \subset S$ of centers. We keep for every point $p \in S \setminus T$ the center in T closest to p, such point is called $neigbour(p)$, and the value of the distance $D(p, neighbour(p))$ is $dist(p)$.
This description of the algorithm concentrate on building the set T of the heads of the clusters. Most of the time is actually spent in updating the invariants, if this is done in a straightforward manner it takes $O(n)$ time per iteration, so in total $O(nk)$ time.

Algorithm 1 Feder and Greene version of furthest-point-heuristic

$T = \emptyset$;
$dist(p) = \infty \quad \forall p \in S$;
while $|T| \leq k$ **do**
  $q = \eta \arg\max\{dist(p) | p \in S \setminus T\}$;
  add q to T;
  update $neighbour(p)$, $dist(p)$ $\forall p \in S \setminus T$
end while

The main result of [FG88] is an efficient way of updating the invariants $neighbour(p)$ and $dist(p)$ when the problem is cast in a d-dimensional Real space and the distance is an L_q metric. In this case overall the furthest point algorithms can be made to run in $O(n \log k)$ time. This result is achieved via the application of hierarchies of bounding boxes inspired by the method of Vaidya for the all-nearest-neighbors problem [Vai89]. Such scheme is rather complex and at the best of my knowledge it has not been implemented.

Sariel Har-Peled in [HP01] solves the Euclidean k-center problem, within an approximation factor 2, for $k = O(\sqrt[3]{n}/\log n)$ in expected time $O(n)$ provided the computational model allows for constant time hashing and floor function. This algorithm uses a clever mix of random sampling, fast point location and the furthest point heuristic.

A version of the algorithm that produces a hierarchy of clusters is described in [Das02].

B. The k-means algorithm (aka Lloyd's algorithm)

Lloyd's algorithm (see [Har75], [For65], [Llo57]) can be seen as an iterative cluster quality booster. It takes as input a rough k-clustering (or more precisely k candidate centers) and produces as output another k-clustering (hopefully of better quality). It is had been shown in [SI84] that the using the sum of squared Euclidean distances as internal quality criterion, the procedure converges to a local minimum for the objective function within a finite number of iterations. Its main building blocks are:

1) Generation of the initial selection of k points;
2) Main iteration loop;
3) Termination condition.

In the main iteration loop, given a set of centroid points, each input point is associated to its closest centroid, and the collection of points associated to a centroid is a cluster. For each cluster a new centroid that is a (weighted) linear combination of the cluster points is recomputed. Thus a new iteration can start.
Given the importance of this algorithm there is a vast literature discussing several shortcomings and improvements to the basic framework. In particular, one well known shortcoming is that some clusters may become empty during the computation.

To overcome this behavior, following [Phi02], we adopt the ISODATA [TG77] technique that splits one of the "largest" clusters so to maintain the number of clusters unchanged.

There exists two basic versions Lloyd's algorithm (also batch k-means) [Llo57] and MacQueen's algorithm (adaptive k-means) [Mac67]. In this second method uses an on-line approach, in which points are added one by one. The centroids are recomputed for each new point that is assigned to a cluster, however points already assigned to clusters do not change when the centroids moves.

It is well known, and our experiments confirm this, that the quality of the initialization has a deep impact on the output quality. Initialization of k-means is a delicate step and several methods are compared in [PLL99], [BF98].

IV. Our contribution

We have produced an upgraded version of the "furthest point algorithm" that exploits the triangular inequality to filter out useless distance computations. This modified algorithm works in any metric space.

Consider in Feder and Greene's algorithm a center $c \in T$ and its associated set of closest points $N(c) = \{p \in S \setminus T \mid neigbour(p) = c\}$. Store such set of point $N(c)$ by decreasing distance to c. When a new center q is selected we scan $N(c)$ in decreasing order of distance and we stop the scan when for a point $p \in N(c)$, we have $D(p,c) \leq D(q,c)/2$. By the triangular inequality, any point p that satisfies this condition cannot be closer to q than to c. This rule filters out form the scan points for which q cannot possibly be an associated center, thus speeding up the update of the invariants. Note that all mutual distances of centers in T must be available, thus, there is a cost $O(k^2)$ to compute and maintain this information. The gain is that potentially fewer than n points need to be scanned at each iteration.

We consider a standard model in which all data reside in main memory. In our scenario the scalability issue arises not because of the need to access slow secondary memories, but because of an excessive number of expensive distance computations. In this context algorithms that as a first step compute all $\Theta(n^2)$ pairwise distances among the input points are ruled out as too expensive[4]. Also methods that perform $\Theta(nk)$ distance computations are too slow except when k is a very small number (e.g. 2 or 3). Using our filtering step within the furthest-point-first algorithms, in terms of asymptotic complexity we could not prove any better bound than the naive $O(nk)$, however experiment have shown that under several different scenarios the speed gain is substantial and this improvement makes the "furthest point algorithm" truly scalable.

[4]Many variants of HAC: Hierarchical Agglomerative Clustering pay this initial high cost

V. Experiments

A. Algorithms and Variants

We made experiments with two variants of the k-center algorithm and three variants of the k-means algorithm. We chose the three initialization methods for k-means that have been amply cited in literature and are relatively simple. More advanced and complex initializations have been ruled out since the possible boost in quality is paid immediately in an excessive time for the initialization, thus falling immediately out of the trade-off region of interest for this study.

- KC This is the k-center algorithm described in Section III-A together with the triangle inequality filtering.
- RS This is the same algorithm as KC but applied to a Random Sample of the input points of size $n' = \sqrt{nk}$, since $k \leq n$ we have $n' \leq n$ always. Afterwards the remaining $n-n'$ input points are associated to the closest center.
- PTS This is the k-means described in section III-B where initial centers are randomly chosen among the input data and the rest of elements are assigned to the closest center [For65].
- RP This is the k-means described in section III-B with an initialization based on Random perturbation of the global centroid point. Points are considered as a distribution with mean μ and standard deviation σ. Centroids are then obtained by generating points along a two-dimensional Gaussian distribution of mean μ and standard deviation σ.
- MQ This is the k-means described in section III-B where seeds are randomly chosen among the input data. The remaining points are assigned one per time to the nearest centroid that must be recomputed [Mac67].

B. Quality measures

As mentioned in section II-A, there are two fundamentally different ways for evaluating the clustering quality: the internal and the external criterion. The first one is based on the evaluation of how the output clustering approx a certain objective function, while the second is based on the comparison between the output clustering and the ground truth. In our experiments the ground truth is always available, so we will use the external measure for evaluating clustering quality.

We denote with $P(S, f_S) = \{C_1, ..., C_k\}$ the ground truth partition formed by a collection of *classes*; and with $\mathcal{O} = \{O_1, .., O_k\}$ the outcome of the clustering algorithm that is a collection of *clusters*. We use three well known quality measure: *F-measure*, *Entropy* and *Accuracy* that has been widely used in information retrieval, (see e.g. [SKK00] [CKVW03] and references therein).

1) F-measure: The F-measure was introduced in [LA99] and is based on the *precision* and *recall* that are concepts well known in the information retrieval literature [Kow97], [vR79]. Given a cluster O_j and a class C_i we have:

$$precision(i,j) = \frac{|C_i \cap O_j|}{|O_j|} \quad recall(i,j) = \frac{|C_i \cap O_j|}{|C_i|},$$

where $||$ denotes the cardinality of a set. Note that precision and recall are real numbers in the range $[0, 1]$. Intuitively precision measures the probability that an element of the class i falls in the cluster j while recall is the probability that an element of the cluster j is also an element of the class i. The F-measure $F(i, j)$ of a cluster j and a class i is the harmonic mean of precision and recall:

$$F(i,j) = 2\frac{precision(i,j)recall(i,j)}{precision(i,j) + recall(i,j)}$$

The F-measure of an entire clustering is computed by the following formula:

$$F = \sum_i \frac{|C_i|}{n} \max_j(F(i,j)),$$

where n is the sum of the cardinality of all the classes. The value of F is in the range $[0..1]$ and a higher value indicates better quality.

2) Entropy: Entropy is a widely used measure in information theory. In a nutshell we can use the relative entropy to measure the amount of uncertainty that we have about the ground truth provided the available information is the computed clustering. Given a cluster O_j and a class C_i, we can define

$$p_{i,j} = \frac{|C_i \cap O_j|}{|C_i|},$$

$$E_j = \sum_i p_{i,j} \log p_{i,j},$$

$$E = \sum_j \frac{|O_j|}{n} E_j,$$

where n is the number of elements of the whole clustering. The value of E is in the range $[0.. \log n]$ and a lower value indicates better quality.

3) Accuracy: While the entropy of a clustering is an average of the entropy of single clusters, a notion of accuracy is obtained using simply the maximum operator:

$$A_j = \max_i p_{i,j}$$

$$A = \sum_j \frac{|O_j|}{n} A_j.$$

The accuracy A is in the range $[0..1]$ and a higher value indicates better quality. We report results on the F-measure, Entropy and Accuracy in our experiments. In general they are all rather consistent. Occasionally when quality indicators give diverging results, we take the F-measure as the most significant one since it balances better the need to attain simultaneously good precision and recall.

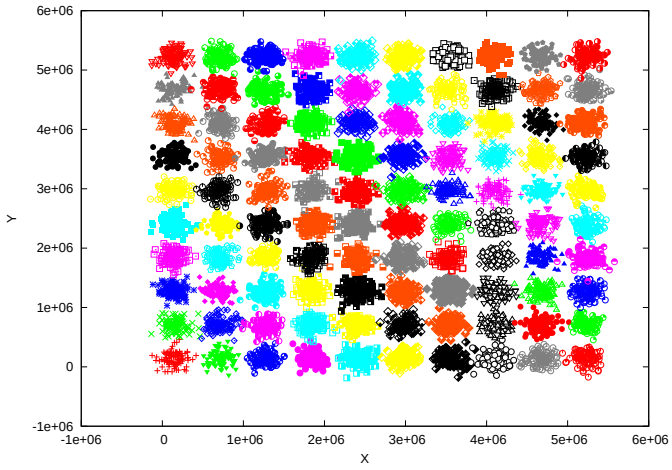

Fig. 1. A graphical representation of BIRCH data set

C. Dataset description

For our experiments we employed two types of data sets: synthetic and real data. Synthetic data consist in a collection of points in the bi-dimensional Cartesian space generated according to the BIRCH [HE02] experiment described in details in section V-D. As real data we used the well known Reuters data set and the resulting snippets of a set of query made to ODP [Pro] (Open Directory Project).

Experiments of synthetic data show that our algorithm is faster than the accelerated K-means [Phi02] and even more accurate. Experiments using real data sets show that both algorithms have comparable quality, but the time cost of k-means is far larger than that of k-center. Especially for on-line snippet clustering, where response time is a crucial parameter this feature could determine the success or failure of the clustering system.

D. Synthetic data

We build synthetic data according to the BIRCH experiment as described in [HE02]. In this experiments the data set is constituted of 10,000 points in a bi-dimensional space. In order to build the clusters we build a 10 x 10 uniform grid in the bi-dimensional space. Each cell of the grid is a square of size $4\sqrt{2}$. The bottom-left vertex of the first cell have coordinates 0,0. The center of each cell is also the center of each cluster. A cluster is populated by inserting 100 points according to a bi-dimensional Gaussian distribution with mean the coordinate of its center and variance 1. Note that according with the Gaussian distribution some points of a cluster can fall in a different cell respect to the one containing its center. In figure 1 we show a graphical representation of the BIRCH data set. Clearly, a distribution like that in the BIRCH experiment (see Figure 1) is not a realistic one, but it is considered a challenging input for clustering algorithms. This experiment gives a hint that the k-center algorithm algorithm can indeed be more competitive than k-means in metric spaces in which the distance function is well defined and there is a high degree of locality.

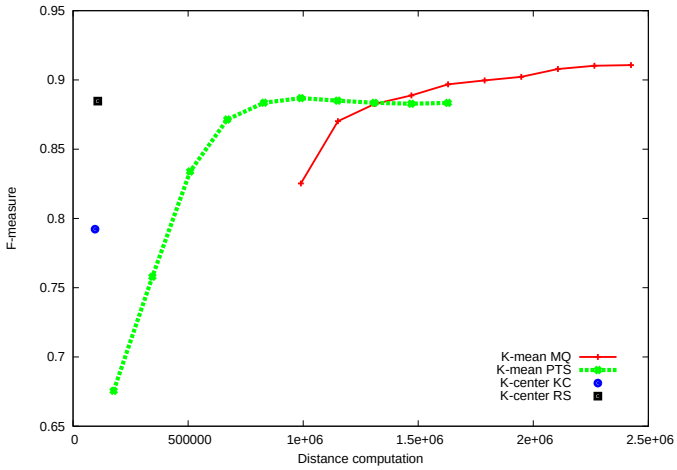

Fig. 2. BIRCH experiment with 10,000 points, 100 clusters. F-measure versus number of distance computations.

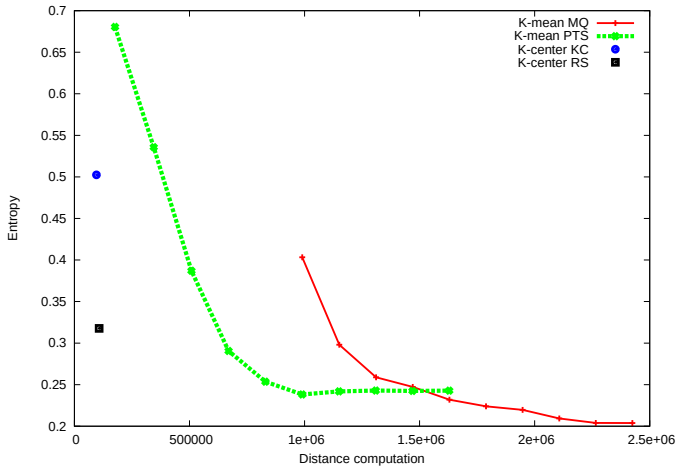

Fig. 3. BIRCH experiment with 10,000 points, 100 clusters. Entropy versus number of distance computations.

In Figures 2, 3 and 4 we show the correlation of the number of inter-point distance computations (x-axis) and the quality of the resulting clustering (y-axis), at each iteration of the k-mean method. Each curve represents the evolution of a variant of the k-means algorithm for each different initialization method. Since k-center is not an iterative algorithm, each variant is represented by a single point.

Algorithm	# Dist.	F-measure	Entropy	Accuracy
K-center	95831	0.792	0.502	0.798
K-center RS	107555	0.884	0.317	0.884
Init. MQ	990000	0.825	0.403	0.837
Init. PTS	176264	0.675	0.680	0.672
Init. RP	149081	0.629	0.834	0.612
K-means MQ	2424365	0.910	0.203	0.915
K-means PTS	1628974	0.883	0.242	0.888
K-means RP	1309060	0.886	0.238	0.888

TABLE I

DATA SET BIRCH WITH 10000 POINTS AND 100 CLUSTERS.

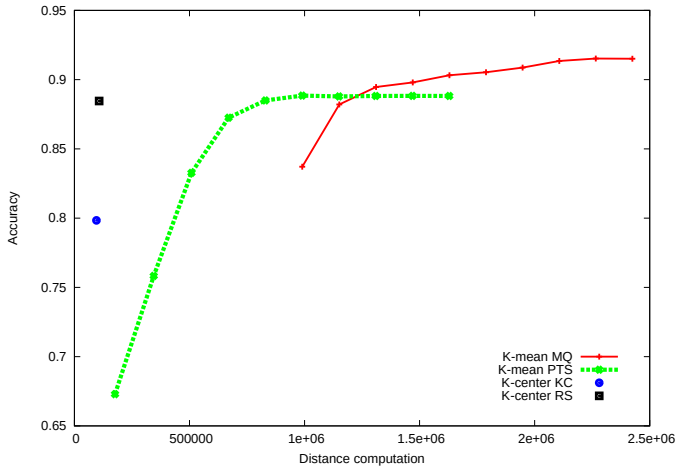

Fig. 4. BIRCH experiment with 10,000 points, 100 clusters. Accuracy versus number of distance computations.

E. Text data

Since the pioneering work of Salton [SM86] it is known that text corpora can be embedded in high dimensional real vector spaces so that properly defined similarity measures for pairs of vectors (e.g. the cosine similarity) approximate well the notion of "similarity of topic" of the two corresponding documents. We tested our algorithm on one quite large static collection of news, the Reuters collection, which is often used as benchmark in the Information Retrieval literature. Also, we tested our algorithm on on-line generated collections of snippets resulting from queries to the ODP (Open Directory Project) hierarchy of topics.

1) Reuters: Reuters is one of the most used data sets in information retrieval [NIS05]. After an initial cleaning to remove multi-labelled documents, we obtain a corpus of 8538 documents of various length organized in 65 mutually exclusive classes. We further remove 10 classes each containing only a document.

In Table II it is shown for each variant of k-center and k-mean (after 10 iterations) the time cost in seconds, the number of distance computations and the quality of the final clustering (using three quality criteria). In terms of number of comparisons there is one order of magnitude difference between k-center and k-means, however in terms of time there is a two orders of magnitude gap. This is due to the fact that the auxiliary operations of book keeping and centroid computation have a large impact.

The evolution of the trade off quality/time at each iteration of k-means is shown in figures 5, 6, and 7.

Table II shows that in terms of time RP and MQ are more expensive k-means initializations than PTS. This is due to the computation of new centroids. In the MQ case we were able to save significantly in time by an incremental computation of new centroids. K-center KC and RS have a cost in time that

Algorithm	# Dist.	Time	F-measure	Entropy	Accuracy
K-center	470782	212	0.340	1.398	0.594
K-center RS	499593	190	0.441	1.240	0.650
Init. MQ	466565	3656	0.479	0.963	0.698
Init. PTS	455365	225	0.263	1.258	0.639
Init. RP	471075	54687	0.182	1.642	0.515
K-means MQ	4630184	44248	0.433	0.948	0.708
K-means PTS	4561664	41868	0.296	0.901	0.742
K-means RP	4483178	97018	0.298	0.924	0.735

TABLE II
DATA SET REUTERS WITH 8528 DOCUMENTS AND 55 CLUSTERS. (TIME IS MEASURED IN SECONDS)

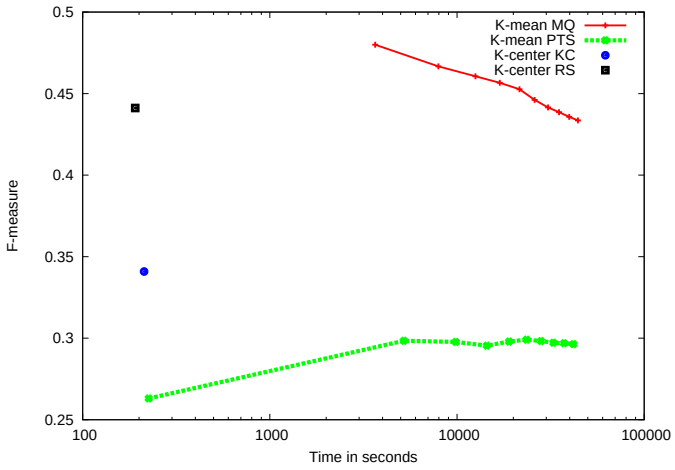

Fig. 5. F-measure of Reuters versus time (in secs) on a logarithmic scale

is two orders of magnitude less than that of K-means MQ and RP. In terms of time cost KC and RS are at a par with PTS, however the quality in terms of F-measure is significantly better than that attained by PTS. After iterating ten times k-means only slightly improves the quality. Note that, in the case of MQ, the F-measure decreases at each iteration of k-means. This is a rare event but a possible one since k-means is guaranteed for certain metrics to reach a local minimum only for the interior quality criterion, not for the exterior one.

2) Web snippets: We made a series of experiments using as input the snippets resulting from a query to the web-based directory "The Open Directory Project" [Pro]. The Open Directory Project (ODP for short) is a pre-classified collection of a few millions of web pages (on December 3, 2003, ODP reached 4M entries) pre-classified into more than 590K categories by a wide group of volunteer human experts. The classification induced by the ODP labelling scheme gives us an objective "ground truth" against which we can compare our clustering. In ODP documents are organized according with a hierarchical ontology. For any snippet we obtain a label for its class by considering only the first two levels of the path on the ODP category tree. For example, if we have two documents, the first one in category **Games** → **Puzzles** → **Anagrams** and the second in category **Games** → **Puzzles** → **Crosswords**, they are both considered in class **Games** →

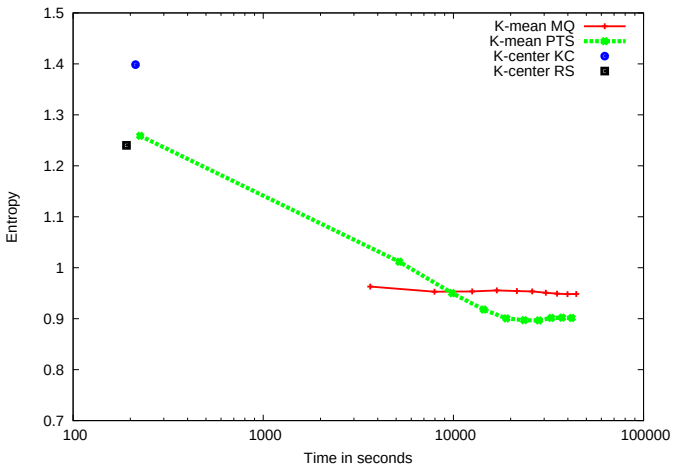

Fig. 6. Entropy of Reuters versus time (in secs) on a logarithmic scale

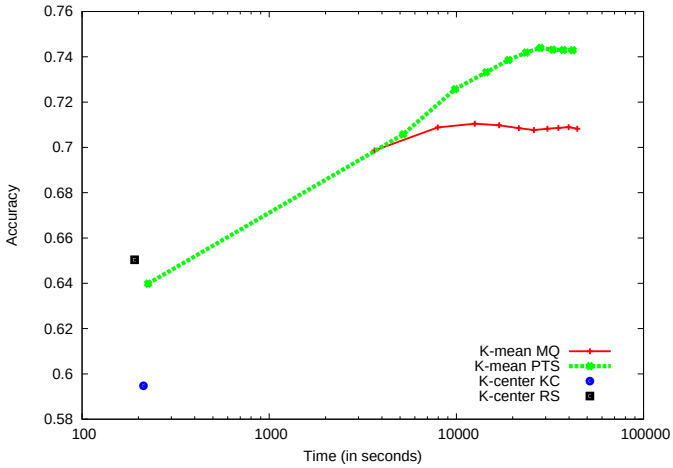

Fig. 7. Accuracy of Reuters versus time (in secs) on a logarithmic scale

Puzzles. This coarsification is needed in order to balance the number of classes and the number of snippets returned by a query. In ODP the textual quality of the snippets is quite variable, therefore to filter out noise in our tests we first collect at most 400 snippets returned from a query process, then we discard snippets that are shorter than 40 characters and containing less then 3 words.

Clustering of snippets is used as an on-line support to web browsing, therefore real-time response is a critical parameter. A clustering phase that introduces a delay comparable to the time needed for just downloading the snippets, thus in effect doubling the user latency, is not acceptable for most users. For this reason, instead of setting a fixed number of iteration to the k-means algorithm, we decided a reasonable time deadline (in our experiments of 5 seconds) and we halt the iterative algorithms at the end of the first iteration that is passed the deadline. We count only the clustering time needed

We planed a set of queries according with the method used in [GDGL05], by dividing queries in three broad families: *ambiguous queries*, *generic queries and specific queries*. For

each query we indicate the number n of snippets found, the number of k of ODP classes, and the time in seconds used to build the TF-IDF model. For each query and each algorithm we indicated the time used for clustering and the quality of the outcome. These results are shown in tables III IV V where we highlight in bold the two best values in each column. We noticed that the quality of the clustering is very dependent on the single query. Moreover, different results could be returned for the same query at different times because of updates made by the ODP team.[5] In terms of time k-center is 5 to 10 times faster than k-means with a quality that is often better than that attained by the k-means variants

Algorithm	Time	F-measure	Entropy	Accuracy
Query = "armstrong" n = 175 k = 52 Build time = 0.692				
KC	**1.258**	0.406	0.885	**0.582**
RS	**1.451**	**0.416**	**0.874**	**0.577**
RP	64.980	0.319	1.154	0.434
MQ	5.808	**0.420**	**0.868**	0.571
PTS	5.759	0.377	0.958	0.525
Query = "jaguar" n = 177 k = 26 Build time = 0.707				
KC	**0.637**	**0.414**	1.042	0.553
RS	**0.691**	0.382	**1.022**	**0.564**
RP	25.982	0.250	1.344	0.389
MQ	6.464	**0.384**	**0.914**	**0.604**
PTS	6.214	0.316	1.113	0.536
Query = "mandrake" n = 137 k = 31 Build time = 0.599				
KC	**0.609**	**0.377**	0.832	**0.693**
RS	**0.725**	**0.404**	**0.669**	**0.722**
RP	29.564	0.246	0.878	0.576
MQ	5.454	0.318	**0.755**	0.664
PTS	5.131	0.299	0.881	0.642
Query = "java" n = 171 k = 46 Build time = 0.783				
KC	**1.188**	0.336	1.111	**0.532**
RS	**1.347**	**0.371**	1.042	**0.520**
RP	60.716	0.267	1.167	0.426
MQ	10.241	**0.343**	**0.984**	**0.520**
PTS	9.999	0.338	**0.975**	**0.520**

TABLE III
CLUSTERING OF SNIPPETS RETURNED BY QUERIES TO THE ODP DIRECTORY FOR *ambuguous queries*.

Algorithm	Time	F-measure	Entropy	Accuracy
Query = "health" n = 154 k = 47 Build time = 0.788				
KC	**1.255**	**0.377**	**0.915**	**0.551**
RS	**1.422**	**0.365**	**0.880**	**0.551**
RP	53.634	0.292	0.971	0.493
MQ	6.958	0.343	0.918	0.519
PTS	5.135	0.334	0.925	0.519
Query = "language" n = 185 k = 33 Build time = 1.022				
KC	**0.956**	0.263	1.321	0.410
RS	**1.029**	**0.298**	**1.301**	**0.432**
RP	44.458	0.255	1.408	0.367
MQ	6.479	**0.280**	**1.298**	**0.432**
PTS	6.038	0.276	1.310	0.421
Query = "machine" n = 186 k = 52 Build time = 0.876				
KC	**1.531**	**0.408**	1.137	**0.483**
RS	**1.689**	0.363	1.125	0.467
RP	83.725	0.322	1.169	0.397
MQ	9.4684	**0.420**	**1.018**	**0.483**
PTS	9.214	0.398	**1.068**	0.467
Query = "music" n = 188 k = 44 Build time = 0.807				
KC	**1.254**	**0.326**	**0.830**	**0.617**
RS	**1.428**	0.300	**0.852**	0.585
RP	57.248	0.239	1.038	0.510
MQ	6.461	**0.314**	0.895	**0.595**
PTS	6.609	0.287	1.004	0.558
Query = "clusters" n = 194 k = 34 Build time = 0.956				
KC	**1.048**	**0.525**	**0.825**	**0.644**
RS	**1.128**	**0.519**	**0.812**	**0.649**
RP	45.496	0.291	1.335	0.391
MQ	6.458	0.465	0.830	0.634
PTS	5.992	0.430	0.872	0.608

TABLE IV
CLUSTERING OF SNIPPETS RETURNED BY QUERIES TO THE ODP DIRECTORY FOR *generic queries*.

VI. PREVIOUS WORK: ALGORITHMS

In most settings problems listed in Section (II-C) are NP-hard for non constant number of clusters k. Thus one usually resorts to heuristics or to approximation algorithms. In this context an approximation algorithm is an algorithm whose output's value is within a bounded multiplicative factor from the value of the optimal value of the corresponding functional.

When k is part of the input parameters, Feder and Greene [FG88] (see also [BE96]) proved that it is NP-hard to approximate the Euclidean k-center problem in 2d within an approximation ratio smaller than 1.822 (for the diameter k-center, smaller than 1.969). Moreover it is NP-hard to approximate the L_1-metric k-center problem in 2d within an approximation ratio smaller than 2.

When k is fixed in [CRW91], [HS91] when the objective function to be optimized is a monotone function of the clusters Euclidean radii (or diameters) then there is a polynomial time algorithm in general with time $O(n^{6k})$. For some special value of k and specific functionals better results can be obtained, e.g. for $k = 2$ the 2-center problem in 2D is solved in time $O(n \log n)$. Inaba, Katoh and Imai [IKI94] observed that for the extended k-means problem one could enumerate all possible decompositions induced by k centers in time $O(n^{kd+1})$ and find thus the optimal solution can be found in polynomial time for fixed k.

A $(1+\epsilon)$-approximation algorithm by Agarwal and Procopiuc [AP98] for the Euclidean k-center problem in R^d runs in time $O(n \log k + (k/\epsilon)^{O(k^{\frac{d-1}{d}})})$, for fixed d and k.

Kalliopoulos and Rao [KR99], improving on previous results of Arora, Raghavan and Rao, give a $(1+\epsilon)$-approximation algorithm for k-median clustering of points in R^d using time $O(2^{1/\epsilon^d} n \log n \log k)$, for fixed d and k.

Matoušek in [Mat00] give a $(1+\epsilon)$-approximation algorithm for k-mean clustering in R^d using time $O(n(\log n)^k (1/\epsilon)^{2k^2 d})$ for fixed d and k.

Har-Peled and Mazudar [HPM04] have improved the results for k-median and k-means by algorithms with complexity linear in n.

[5]For sake of replicating the experiments, when we make a new query we also cache its results.

Algorithm	Time	F-measure	Entropy	Accuracy
Query = "mickey mouse" n = 66 k = 26 Build time = 0.328				
KC	**0.284**	**0.549**	0.806	**0.606**
RS	**0.354**	**0.609**	**0.666**	**0.666**
RP	7.516	0.473	0.816	0.530
MQ	5.267	0.437	0.833	0.530
PTS	3.403	0.466	**0.760**	0.575
Query = "olympic games" n = 183 k = 41 Build time = 0.939				
KC	**1.206**	**0.335**	1.028	**0.568**
RS	**1.334**	**0.331**	**0.971**	**0.551**
RP	40.537	0.258	1.088	0.502
MQ	10.695	0.291	1.006	0.519
PTS	10.516	0.301	**0.967**	0.535
Query = "steven spielberg" n = 73 k = 11 Build time = 0.262				
KC	**0.119**	**0.646**	**0.670**	**0.753**
RS	**0.130**	**0.652**	**0.639**	**0.726**
RP	3.665	0.387	1.109	0.493
MQ	1.428	0.513	0.673	**0.726**
PTS	1.357	0.515	0.809	0.657

TABLE V
CLUSTERING OF SNIPPETS RETURNED BY QUERIES TO THE ODP DIRECTORY FOR *specific queries*.

A second approach to the problem is to develop algorithms with some theoretical guarantee but also simple enough to be of practical value. In [KMN+02b] it is shown a $(9+\epsilon)$-approximation for the k-means problem in Euclidean d-dimensional space, based on local search for moves improving the value of the objective function. An hybrid algorithm merging Lloyd's algorithm with the the local search technique is also tested for assessing empirical performance. This local search technique has been applied to the Euclidean k-median problem in [KPR98], [CG99], [AGK+01], and to the metric k-means problem in [MP02].

Associating points to clusters via nearest-neighbor computations is the most expensive task in Lloyd's algorithm and number of speedup techniques have been proposed recently [ARS98], [KMN+02a], [PM99], [PM00], [Phi02]. Some are based on using kd-trees for speeding up nearest neighbor search [ARS98], [PM99], [KMN+02a] and these can be used in Euclidean spaces of low dimension (say up to 5 or 6). Others like [Phi02], [PM00], [Elk03] are based on using the triangular inequality and work in metric spaces.

Other approaches which we will not cover use simulated annealing, branch-and-bound searching, gradient descent, genetic algorithms for which no provable approximation bounds are known.

Mishra et al. [MOP01] show that, for the k-median problem (both in the discrete and continuous setting), when feeding a clustering algorithm with a random sample of the input data, the obtained solutions is of comparable quality with respect to using all of the input data. A similar sample-based result is shown in [ADPR00] for the k-center problem. In [GMMO00] a similar input reduction effect is obtained via a divide and conquer strategy.

For the metric k-median problem Indyk in [Ind99] shows that, given as a subroutine a $O(n^2)$ algorithm that produces βk clusters within a factor α of the cost of the optimal k-clustering, it is possible via appropriate sampling to attain an $O(kn)$ expected time algorithm producing 2β clusters within a factor $3(2+\alpha)$ of the cost of the optimal k-clustering, with high probability.

Sometimes the problem is cast in a specific computational model such as: the incremental setting, with points added one by one to the data structure [CCFM97], [Can93], the parallel setting [Dat94], the data stream setting .

In Model based clustering one assumes that the data are generated by sampling a mixture of density functions from a given class. The objective is to find the parameters of these functions that better fit the data.

VII. PREVIOUS WORK: INFORMATION RETRIEVAL

One of the applicative areas fields in which the model we refer to finds a natural example is that of Information retrieval, in particular in regarding problem of clustering documents that are mapped to points in a very high dimensional vector space [SM86][DS05] [DFG01].

Often in the context of IR application the clustering problem is reformulated in terms of "similarity" functions instead of distance functions [DM01][DGK02].

Texts related to the WWW (pages, snippets) have recently received much attention [SGM00], [FG04], [HGKI02]. Often in this context techniques based on text are combined with techniques based on the hyper-textual graph structure of the WWW [MS00].

Clustering techniques have been used to improve the quality of wide-topic searching on the web [CPKT92], [CKP93], [HP96], [HGI00], [ZE98] .

VIII. CONCLUSIONS

The K-means algorithm (Lloyd's algorithm) is the a well known method for clustering, used in a wide range of applications, whose pros and cons are well known. In contrast, the furthest-point-first method for k-clustering has been considered so far only of theoretical interest. In this paper we show the result of experiments in the area of Information Retrieval indicating that the furthest-point-first method, with additional filtering steps, is capable of producing high quality clusters within a fraction of the time used by improved versions of k-means. In particular we dispense with the need to compute centroids, thus we can deal easily with situations where the notion of a centroid is not a natural or well defined one, or it is expensive to compute and update.

REFERENCES

[ABKY88] T. Asano, B. Bhattacharya, M. Keil, and F. Yao. Clustering algorithms based on minimum and maximum spanning trees. In *SCG '88: Proceedings of the fourth annual symposium on Computational geometry*, pages 252–257. ACM Press, 1988.

[ADPR00] Noga Alon, Seannie Dar, Michal Parnas, and Dana Ron. Testing of clustering. In *Proc. 41st Annual Symposium on Foundations of Computer Science*, pages 240–250. IEEE Computer Society Press, Los Alamitos, CA, 2000.

[AGK+01] Vijay Arya, Naveen Garg, Rohit Khandekar, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *ACM Symposium on Theory of Computing*, pages 21–29, 2001.

[And73] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, 1973.

[AP98] Pankaj K. Agarwal and Cecilia Magdalena Procopiuc. Exact and approximation algorithms for clustering (extended abstract). In *Symposium on Discrete Algorithms*, pages 658–667, 1998.

[ARS98] K. Alsabti, S. Ranka, and V. Singh. An efficient k-means clustering algorithm. In *IPPS/SPDP Workshop on High Performance Data Mining*, 1998.

[BE96] Marshall Wayne Bern and David Eppstein. Approximation algorithms for geometric problems. In Dorit Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, chapter 8, pages 296–345. PWS Publishing, 1996.

[Ber02] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[BF98] Paul S. Bradley and Usama M. Fayyad. Refining initial points for K-Means clustering. In *Proc. 15th International Conf. on Machine Learning*, pages 91–99. Morgan Kaufmann, San Francisco, CA, 1998.

[Can93] Fazli Can. Incremental clustering for dynamic information processing. *ACM Trans. Inf. Syst.*, 11(2):143–164, 1993.

[CCFM97] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *STOC*, pages 626–635, 1997.

[CG99] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *IEEE Symposium on Foundations of Computer Science*, pages 378–388, 1999.

[CKP93] Douglass R. Cutting, David Karger, and Jan Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 126–135, 1993.

[CKVW03] D. Cheng, R. Kannan, S. Vempala, and G. Wang. On a recursive spectral algorithm for clustering from pairwise similarities. Technical Report MIT-LCS-TR-906, MIT LCS, 2003.

[CPKT92] Douglass R. Cutting, Jan O. Pedersen, David Karger, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.

[CRW91] Vasilis Capoyleas, Günter Rote, and Gerhard J. Woeginger. Geometric clusterings. *J. Algorithms*, 12(2):341–356, 1991.

[Das02] Sanjoy Dasgupta. Performance guarantees for hierarchical clustering. In *COLT '02: Proceedings of the 15th Annual Conference on Computational Learning Theory*, pages 351–363. Springer-Verlag, 2002.

[Dat94] Amitava Datta. Efficient parallel algorithms for geometric k-clustering problems. In P. Enjalbert, E.W. Mayr, and K.W. Wagner, editors, *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science, STACS'94 (Caen, France, February 24-26, 1994)*, volume 775 of *LNCS*, pages 475–486, Berlin, 1994. Springer-Verlag.

[DF85] M. Dyer, , and A. Frieze. A simple heuristic for the p-center problem. *Oper. Res. Lett.*, 3(6):285–288, 1985.

[DFG01] I. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In R. Grossman, G. Kamath, and R. Naburu, editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, 2001.

[DFK+99] Drineas, Frieze, Kannan, Vempala, and Vinay. Clustering in large graphs and matrices. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1999.

[DGK02] Inderjit S. Dhillon, Yuqiang Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, page 131. IEEE Computer Society, 2002.

[Dhi01] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, pages 269–274, 2001.

[DM01] Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Mach. Learn.*, 42(1-2):143–175, 2001.

[DO74] B.S. Duran and P.K. Odell. *Cluster analysis–A survey*. Springer-Verlag, 1974.

[DS05] Franca Debole and Fabrizio Sebastiani. An analysis of the relative hardness of Reuters-21578 subsets. *Journal of the American Society for Information Science and Technology*, 2005. Forthcoming.

[Elk03] Charles Elkan. Using the triangle inequality to accelerate k-means. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 147–153, 2003.

[FG88] Tomas Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 434–444. ACM Press, 1988.

[FG04] Paolo Ferragina and Antonio Gulli. The anatomy of snaket: A hierarchical clustering engine for web-page snippets. In *PKDD: Knowledge Discovery in Databases: PKDD 2004, 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 506–508, 2004.

[FMW00] Sándor P. Fekete, Joseph S. B. Mitchell, and Karin Weinbrecht. On the continuous weber and k -median problems (extended abstract). In *Symposium on Computational Geometry*, pages 70–79, 2000.

[For65] E. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21(3):768–780, 1965.

[GDGL05] Emilio Di Giacomo, Walter Didimo, Luca Grilli, and Giuseppe Liotta. A topology-driven approach to the design of web meta-search clustering engines. In *31st Annual Conference on Current Trends in Theory and Practice of Informatics (SOFSEM)*. Springer-Verlag, 2005.

[GMMO00] Sudipto Guha, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams. In *IEEE Symposium on Foundations of Computer Science*, pages 359–366, 2000.

[Gon85] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

[GRS04] Shobha Ganapathy, C. Ranganathan, and Balaji Sankaranarayanan. Visualization strategies and tools for enhancing customer relationship management. *Commun. ACM*, 47(11):92–99, 2004.

[Har75] John A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, Inc., 1975.

[HE02] Greg Hamerly and Charles Elkan. Alternatives to the k-means algorithm that find better clusterings. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 600–607. ACM Press, 2002.

[HGI00] Taher H. Haveliwala, Aristides Gionis, and Piotr Indyk. Scalable techniques for clustering the web. In *WebDB (Informal Proceedings)*, pages 129–134, 2000.

[HGKI02] Taher H. Haveliwala, Aristides Gionis, Dan Klein, and Piotr Indyk. Evaluating strategies for similarity search on the web. In *WWW*, pages 432–442, 2002.

[HP96] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 76–84, Zürich, CH, 1996.

[HP01] Sariel Har-Peled. Clustering motion. In *IEEE Symposium on Foundations of Computer Science*, pages 84–93, 2001.

[HPM04] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300. ACM Press, 2004.

[HPS05] Sariel Har-Peled and Bardia Sadri. How fast is the k-means method? *Algorithmica*, 41(3):185–202, 2005.

[HS85] D. S. Hochbaum and D. B. Shmoys. A best possible approximation algorithm for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[HS91] S.L. Hakimi and E.F. Schmeichel. Fitting polygonal functions to a set of points in the plane. *Graphical Models and Image Processing*, 53:132–136, 1991.

[IKI94] Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted voronoi diagrams and randomization to variance-based k -clustering (extended abstract). In *Symposium on Computational Geometry*, pages 332–339, 1994.

[Ind99] Piotr Indyk. Sublinear time algorithms for metric space problems. In *STOC*, pages 428–434, 1999.

[JD88] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[KMN+02a] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):881–892, 2002.

[KMN+02b] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. In *SCG '02: Proceedings of the eighteenth annual symposium on Computational geometry*, pages 10–18. ACM Press, 2002.

[Kow97] Gerald Kowalski. *Information Retrieval Systems: Theory and Implementation*. Boston Kluwer Academic Publishers, 1997.

[KPR98] Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. In *SODA '98: Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 1–10. Society for Industrial and Applied Mathematics, 1998.

[KR99] Stavros G. Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the euclidean kappa-median problem. In *ESA*, pages 378–389, 1999.

[KVV00] R. Kannan, S. Vempala, and A. Veta. On clusterings-good, bad and spectral. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 367. IEEE Computer Society, 2000.

[LA99] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22. ACM Press, 1999.

[Llo57] S.P. Lloyd. Least squares quantization in pcm. Technical report, Bell Laboratories Technical Note, 1957. Eventually published in IEEE Trans. on Inform. Thy. IT-28(2), 1982.

[Mac67] MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings Fifth Berkeley Symposium on Math. Stat. and Prob.*, pages 281–297. University of California Press, 1967.

[Mat00] Jirí Matousek. On approximate geometric k-clustering. *Discrete & Computational Geometry*, 24(1):61–84, 2000.

[MOP01] Nina Mishra, Dan Oblinger, and Leonard Pitt. Sublinear time approximate clustering. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 439–447, 2001.

[MP02] Ramgopal R. Mettu and C. Greg Plaxton. Optimal time bounds for approximate clustering. In *UAI '02, Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence, University of Alberta, Edmonton, Alberta, Canada, August 1-4, 2002*, pages 344–351, 2002.

[MS00] Dharmendra S. Modha and W. Scott Spangler. Clustering hypertext with applications to web searching. In *ACM Conference on Hypertext*, pages 143–152, 2000.

[NIS05] NIST. http://trec.nist.gov/data/reuters/reuters.html, 2005.

[Phi02] Steven J. Phillips. Acceleration of k-means and related clustering algorithms. In *ALENEX*, pages 166–177, 2002.

[PLL99] José Manuel Peña, Jose Antonio Lozano, and Pedro Larrañaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040, 1999.

[PM99] Dan Pelleg and Andrew Moore. Accelerating exact k -means algorithms with geometric reasoning. In *ACM SIGKDD International Confrence on Knowledge Discovery and Data Mining*, pages 277–281, 1999.

[PM00] Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734, 2000.

[Pro] Open Directory Project. http://dmoz.org/.

[SGM00] Alexander Strehl, Joydeep Ghosh, and Raymond J. Mooney. Impact of similarity measures on web-page clustering. In *Proc. AAAI Workshop on AI for Web Search (AAAI 2000), Austin*, pages 58–64. AAAI/MIT Press, July 2000.

[SI84] S.Z. Selim and M.A. Ismail. K-means type algorithms: a generalized convergence theorem and characterization of local optimality. *IEEE Trans. Pattern Anal. Mach. Inteli*, 6:81–87, 1984.

[SKK00] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *In KDD Workshop on Text Mining,2000.*, 2000.

[SM86] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

[TG77] J. T. Tou and R. C. Gonzalez. *Pattern Recognition Principles*. Addison-Wesley, Reading, MA, 1977.

[Vai89] Pravin M. Vaidya. An O(n log n) algorithm for the all-nearest-neighbors problem. *Discrete & Computational Geometry*, 4:101–115, 1989.

[vR79] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

[ZE98] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *SIGIR*, pages 46–54, 1998.

[Zup82] J. Zupan. *Clustering of Large Data Sets*. Chemometrics Research Studies Ser. Research Studies Press, Chichester, 1982.