COOLCAT: An entropy-based algorithm for categorical clustering *

Daniel Barbará *Julia Couto* *Yi Li*
George Mason University
Information and Software Engineering Department
Fairfax, VA 22030
Phone: 703-9931627, Fax=703-9931638
{dbarbara,jics,yli1}@gmu.edu

October 1, 2001

Abstract

Clustering of categorical attributes is a difficult problem that has not received as much attention as its numerical counterpart. In this paper we explore the connection between clustering and entropy: clusters of similar points have lower entropy than those of dissimilar ones. We use this connection to design a heuristic algorithm, COOLCAT, which is capable of efficiently cluster large data sets of records with categorical attributes. In contrast with other categorical clustering algorithms published in the past, COOLCAT's clustering results are very stable for different sample sizes and parameter settings. Also, the criteria for clustering is a very intuitive one, since it is deeply rooted on the well-known notion of entropy. We demonstrate the efficiency and scalability of COOLCAT by a series of experiments on real and synthetic data sets.

1 Introduction

Clustering is a widely used technique in which data points are partitioned into groups, in such a way that points in the same group, or cluster, are more similar among themselves than to those in other clusters. Clustering of categorical attributes (i.e., attributes whose domain is not numeric) is a difficult, yet important task: many fields, from statistics to psychology deal with categorical data. In spite of its importance, the task of categorical clustering has received scant attention in the KDD community as of late, with only a handful of publications addressing the problem ([15, 12, 10]).

Much of the published algorithms to cluster categorical data rely on the usage of a distance metric that captures the separation between two vectors of categorical attributes, such as the Jaccard coefficient [26]. In this paper, we present COOLCAT (the name comes from the fact that we reduce the entropy of the clusters, thereby "cooling" them), a novel method which uses the notion of entropy to group records. We argue that a classical notion such as entropy is a more natural and intuitive way of relating records, and more importantly does not rely in arbitrary distance metrics.

This paper is set up as follows. Section 2 offers the background and relationship between entropy and clustering, and formulates the problem. Section 3 reviews the related work. Section 4 describes COOLCAT, our algorithm. Section 5 presents the experimental evidence that demonstrates the advantages of COOLCAT. Finally, Section 6 presents conclusions and future work.

*This work has been supported by NSF grant IIS-9732113

2 Background and problem formulation

In this section, we present the background of entropy and clustering and formulate the problem.

2.1 Entropy and Clustering

Entropy is the measure of information and uncertainty of a random variable [24]. Formally, if X is a random variable, $S(X)$ the set of values that X can take, and $p(x)$ the probability function of X, the entropy $E(X)$ is defined as shown in Equation 1.

$$E(X) = -\sum_{x \in S(X)} p(x)log(p(x)) \quad (1)$$

The entropy of a multivariate vector $\hat{x} = \{X_1, \cdots, X_n\}$ can be computed as shown in Equation 2.

$$E(\hat{x}) = -\sum_{x_1 \in S(X_1)} \cdots \sum_{x_n \in S(X_n)} p(x_1, \cdots, x_n)logp(x_1, \cdots, x_n) \quad (2)$$

Entropy is sometimes referred to as a measure of the amount of "disorder" in a system. A room with socks strewn all over the floor has more entropy than a room in which socks are paired up, neatly folded, and placed in one side of your sock and underwear drawer.

2.2 Problem formulation

The problem we are trying to solve can be formulated as follows. Given a data set D of N points $\hat{p_1}, \cdots, \hat{p_N}$, where each point is a multidimensional vector of d categorical attributes, i.e., $\hat{p_j} = (p_j^1, \cdots, p_j^d)$, and given an integer k, we would like to separate the points into k groups $C_1, \cdots, C_k$, or clusters, in such a way that we minimize the entropy of the whole arrangement. Unfortunately, this problem is NP-Complete, and moreover, difficult to approximate [11]. In fact, the problem is NP-Complete for *any* distance function $d(x, y)$, defined over pairs of points x, y, such that the function maps pairs of points to real numbers (and hence, our entropic function qualifies), therefore we need to resort to heuristics to solve it.

We first have to resolve the issue of what we mean by the "whole entropy of the system." In other words, we have to make our objective function clear. We aim to minimize the expected entropy, whose expression is shown in Equation 3, where $E(P(C_1)), \cdots, E(P(C_k))$, represent the entropies of each cluster, $P(C_i)$ denotes the points assigned to cluster C_i, $P(C_i) \subset D$, with the property that $P(C_i) \cap P(C_j) = \emptyset$, for all $i, j = 1, .., k \;\; i \neq j$. The symbol $\breve{C} = \{C_1, \cdots, C_k\}$ represents the clustering.

$$\bar{E}(\breve{C}) = \sum_k (\frac{|P(C_k)|}{|D|}(E(P(C_k)))) \quad (3)$$

This function, as we will see later, allows us to implement an incremental algorithm that can effectively deal with large datasets, since we do not need to look at the entire set of points to decide about the entropy of an arrangement. Rather, we will be able to decide for each point, how it would affect the entropy of each of the existing clusters if placed in each one of them.

The solution we propose in this paper (and present in Section 4) is a heuristic based in finding a set of initial clusters (using the entropic criteria), and then incrementally (greedily) add points to the clusters according to a criteria that minimizes Equation 3.

Cluster #	Clustering 1		Clustering 2		Clustering 3	
	members	E	members	E	Members	E
Cluster0	{"*red*","*heavy*"} {"*red*","*medium*"}	1.0	{"*red*","*heavy*"} {"*blue*","*light*"}	2.0	{"*red*,*heavy*"}	0
Cluster1	{"*blue*","*light*"}	0	{"*red*","*medium*"}	0	{"*red*","*medium*"} {"*blue*","*light*"}	2.0
Exp.E		0.66		1.33		1.33

Figure 1: Three different clusterings for the set v_1, v_2, v_3. Clustering 1 minimizes the expected entropy of the two clusters.

Furthermore, we make a simplification in the computation of entropy of a set of records. We assume independence of the attributes of the record, transforming Equation 2 into Equation 4. In other words, the joint probability of the combined attribute values becomes the product of the probabilities of each attribute, and hence the entropy can be calculated as the sum of entropies of the attributes.

$$\begin{aligned} E(\hat{x}) &= -\sum_{x_1 \in S(X_1)} \cdots \sum_{x_n \in S(X_n)} (p(x_1)\cdots p(x_n)) log(p(x_1)\cdots p(x_n)) \\ &= E(X_1) + E(X_2) + \cdots + E(X_n) \end{aligned} \tag{4}$$

Assume that we have a set of three records, v_1, v_2 as before, and v_3 = {"*red*","*medium*"}, and we want to form two clusters with them. Figure 1 shows all the possible arrangements, with the entropy of each cluster, and the expected entropy in each arrangement. As we can see, the minimum expected entropy is that of arrangement 1, which obviously is the correct way of clustering the records (using two clusters).

However, in the cases we can demonstrate that there is a correlation between two or more attributes of the data set, we will change the data points by creating attributes that reflect these correlations and then apply Equation 4 to compute the join entropy. For instance, if the data set is composed of records of attributes A, B, C, D, E, F and we know that A, B, A, C and E, F are correlated. we will convert the data set into one having records with attributes AB, AC, D, EF and compute the entropy assuming that these new attributes are independent. Notice that for the grouped attributes, we are in effect computing their joint probabilities.

2.3 Entropy and the similarity coefficients

As we shall prove shortly, as a measure of the similarity between two vectors, the use of entropy is equivalent to that of other widely-used similarity coefficients [26]. Similarity coefficients are best explained by the 2x2 association table shown in Figure 2, where 1 refers to the presence of a variable and 0 to its absence.

The simple matching coefficient (SM) is defined as shown in Equation 5. Notice that this coefficient takes into account the joint absence of a variable (as indicated by d in the table).

$$SM = \frac{(a+d)}{(a+b+c+d)} \tag{5}$$

	1	0
1	a	b
0	c	d

Figure 2: Association table: 1 refers to the presence of a variable, and 0 to its absence.

The Jaccard (J) coefficient is shown in Equation 6. It avoids the use of joint absences of a variable in the calculation of similarity.

$$J = \frac{a}{(a+b+c)} \tag{6}$$

Lemma 1 *For any vectors p, q, u, v in a data set, $SM(\{p,q\}) = SM(\{u,v\})$* **iff** *$E(\{p,q\}) = E(\{u,v\})$.*

Proof: Throughout the proof, we will use $SM(\{p,q\} = \frac{a_1+d_1}{a_1+b_1+c_1+d_1}$ and $SM(\{u,v\}) = \frac{a_2+d_2}{a_2+b_2+c_2+d_2}$ where a_1, b_1, c_1, d_1 are the entries of the association table of p, q, and a_2, b_2, c_2, d_2 those for the association table of u, v. Notice that since the vectors are drawn from the same data set, they all have the same number of attributes, therefore, $a_1+b_1+c_1+d_1 = a_2+b_2+c_2+d_2$. We shall proceed with the proof of the two implications.

Part a) If $SM(\{p,q\}) = SM(\{u,v\})$, then $E(\{p,q\}) = E(\{u,v\})$:

Since $SM(\{p,q\}) = SM(\{u,v\}) = k$, then $a_1+d_1 = k(a_1+b_1+c_1+d_1)$ and $a_2+d_2 = k(a_2+b_2+c_2+d_2)$. Also, since $a_1+b_1+c_1+d_1 = a_2+b_2+c_2+d_2$, it follows that $a_1+d_1 = a_2+d_2$, and $b_1+c_1 = b_2+c_2$. Now, since for matching attributes, the entropy contribution will be 0, and for non-matching attributes, the entropy contribution will be $(2\frac{1}{2}log(\frac{1}{2}))$, then it follows that $E(\{p,q\}) = (b_1+c_1)log(\frac{1}{2}) = (b_2+c_2)log(\frac{1}{2}) = E(\{u,v\})$.

Part b) If $E(\{p,q\}) = E(\{u,v\})$, then $SM(\{p,q\}) = SM(\{u,v\})$: Since for any pair of vectors, the matching attributes will not contribute to the total entropy, while the non-matching attributes will contribute $log(\frac{1}{2})$ (i.e., an equal amount per attribute), then $E(\{p,q\}) = E(\{u,v\})$ implies that $b_1+c_1 = b_2+c_2$. Now, since $a_1+b_1+c_1+d_1 = a_2+b_2+c_2+d_2$, it follows that $a_1+d_1 = a_2+d_2$ proving the implication.$\diamond$

Lemma 2 *For any vectors p, q, u, v in a data set, $SM(\{p,q\}) > SM(\{u,v\})$* **iff** *$E(\{p,q\}) < E(\{u,v\})$.*

Proof: Part a) If $SM(\{p,q\}) > SM(\{u,v\})$ then $E(\{p,q\}) < E(\{u,v\})$.

Since the first coefficient is larger than the second, it means that more attribute values agree for the two vectors ($a_1+d_1 > a_2+d_2$). This, obviously reduces the entropy of the pair, since agreeing values contribute with 0 to the final entropy count.

Part b) If $E(\{p,q\}) < E(\{u,v\})$, then $SM(\{p,q\}) > SM(\{u,v\})$.

Since the entropy of the first pair is less than that of the second, it means that less attributes in the vectors disagree (since those contribute with the value $log(\frac{1}{2})$), i.e., $b_1+c_1 < b_2+c_2$. Therefore, the coefficient will be larger (more values agree). $\diamond$

Lemma 3 *For any vectors p, q, u, v in a data set, $SM(\{p,q\}) < SM(\{u,v\})$* **iff** *$E(\{p,q\}) > E(\{u,v\})$.*

Proof: Similar to Lemma 2. $\diamond$

Theorem 1 *Given two similarity matrices built over the same data set, one using SM and the*

	1	0
1	1	1
0	1	0

Figure 3: Association table for vectors $v_1 = 1, 1, 0$ and $v_3 = 1, 0, 1$.

other entropy, if we order the first matrix in descending order of entries, and the second in ascending order, the relative ordering of the pairs of points is the same in both cases.

Proof: By Lemmas 1, 2 and 3. $\diamond$

As a consequence of Theorem 1, in any algorithm that uses the SM coefficient as a similarity measure, we could also use entropy as well, obtaining identical results. For instance, ROCK [15] (which we shall review in Section 3) uses an "extended" similarity measure in which the coefficient is computed as the number of attributes that agree divided by the total number of attribute values, calling this coefficient the Jaccard coefficient. (For instance, in the example shown in Section 2.2, the coefficient $J(\{v_1, v_2\}) = 0$, since no values are the same in $v_1 = < red, heavy >$ and $v_2 = < blue, light >$; but $J(\{v_1, v_3\}) = \frac{1}{3}$, since one value out of three is the same in the pair $v_1 = < red, heavy >$, $v_3 = \{"red", "medium"\}$.) In fact, it is easy to see that if we convert the "*red*", "*heavy*", and "*medium*" values into binary variables, the vectors become $v_1 = 1, 1, 0$ and $v_3 = 1, 0, 1$, and the association table for v_1, v_3 is the one shown in Figure 3. Then the SM coefficient can be easily computed as $\frac{1}{3}$. The Jaccard coefficient in this case would also be the same, as $d = 0$. This fact is true for every case in which we convert the values into binary variables since we will never have a column in which the two binary values are 0. So, it is easy to see that Theorem 1 will always apply when the Jaccard coefficient is applied to nominal attributes by converting the values to binary variables, since then the Jaccard coefficient and the SM coefficient would be equal.

Our algorithm, COOLCAT, however, uses the fact that entropy, unlike the SM or Jaccard coefficients, can serve as a measure of similarity among any set of vectors (not just two).

2.4 Expected entropy and the Minimum Description Length principle

The Minimum Description Length principle (MDL) [22, 23] recommends choosing the model that minimizes the number of bits needed to encode it. This principle is widely used to compare classifiers (see [20]) but it has not been used much to deal with clustering. (An idea of how to do this by encoding the clusters centers is sketched in [28]).

An optimal encoding for a set of clusters can be realized by assigning codes to each attribute of the data based on the probability that the attribute appears on the cluster, using a Hoffman code. This encoding will have length $\sum_i P(A_i = V_{ij}) log P(A_i = V_{ij})$ for each cluster C_k. So, it can be minimized by minimizing the expected value of that function. But, this is precisely the function we have selected to minimize: the expected entropy of the clustering. So, our technique aims to find the clustering that follows the MDL principle. This has important implications: the fact that the encoding of the clusters is minimal implies that we can expect very concise representations of the clusters we have found at any given point. This in turn makes possible the incremental processing of further data points without having to keep all the previous data points in memory, but just the concise representation of the clusters they form.

$\{1,2,3\}$	$\{1,2,4\}$	$\{1,2,5\}$	$\{1,3,4\}$	$\{1,3,5\}$	$\{1,4,5\}$	$\{2,3,4\}$
$\{2,3,5\}$	$\{2,4,5\}$	$\{3,4,5\}$	$\{1,2,6\}$	$\{1,2,7\}$	$\{1,6,7\}$	$\{2,6,7\}$

Figure 4: A market basket data set

	Solution				
Cluster1	$\{1,2,3\}$	$\{1,2,4\}$	$\{1,2,5\}$	$\{1,3,4\}$	$\{1,3,5\}$
	$\{1,4,5\}$	$\{2,3,4\}$	$\{2,3,5\}$	$\{2,4,5\}$	$\{3,4,5\}$
Cluster2	$\{1,2,6\}$	$\{1,2,7\}$	$\{1,6,7\}$	$\{2,6,7\}$	

Figure 5: The clustering solution obtained by ROCK and by expected entropy for the data of Figure 4

2.5 Evaluating clustering results

This section illustrates the difficulties in evaluating the results of clustering algorithms. A frequent problem one encounters when applying clustering algorithms in practice is the difficulty in evaluating the solutions. Different clustering algorithms (and sometimes multiple applications of the same algorithm using slight variations of initial conditions or parameters) result in very different solutions, all of them looking plausible. This stems from the fact that there is no unifying criteria to define clusters, and more often than not, the final clusters found by the algorithm are in fact the ones that correspond to the criteria used to drive the algorithm.

To illustrate this point, let us consider the following example, taken from [15], a paper that describes ROCK, a categorical clustering algorithm by Guha, Rastogi and Shim. Figure 4 shows a series of transactions that represent "market baskets" of products acquired by clients in a shop. (Each product represented by an integer.) The problem is to cluster these baskets into two clusters. Figure 5 shows both the solution obtained by ROCK [1] and also by finding the minimum expected entropy (both solutions are identical).

How do we know this is a good solution? Authors have pondered about good ways to validate clusters found by algorithms (e.g., see [17, 1, 6]). Two widely used methods are the following:

- *Significance Test on External Variables* This technique calls for the usage of significance tests that compare the clusters on variables not used to generate them. One way of doing this is to compute the entropy of the solution using a variable that did not participate in the clustering. (A class attribute.) The entropy of an attribute C in a cluster C_k is computed as shown in Equation 7, where V_i denotes one of the possible values that C can take. The evaluation is performed by computing the expected entropy (taken into consideration the cluster sizes).

$$E(C_k) \;=\; \sum_i P(C \;=\; V_i) log P(C \;=\; V_i) \tag{7}$$

- *The category utility function* The category utility (CU) function [13] attempts to maximize both the probability that two objects in the same cluster have attribute values in common and

[1] ROCK, as we shall explain in Section 3, uses a combination of a Jaccard coefficient [26] and computation of common neighbors among points to drive the clustering algorithm.

the probability that objects from different clusters have different attributes. The expression to calculate the expected value of the CU function is shown in Equation 8.

$$\bar{CU} = \sum_k \frac{\|C_k\|}{|D|} \sum_i \sum_j [P(A_i = V_{ij}/C_k)^2 - P(A_i = V_{ij})^2] \quad (8)$$

We have used both techniques in validating our results, as shall be seen in the experimental section.

3 Related Work

Clustering is an extensively researched area not only by data mining and database researchers [21, 8, 29, 9, 14, 15, 2], but also by people in other disciplines [7, 26]. Most of the efforts, however, have been focused in clustering of numerical data records.

Among the numerical clustering algorithms, ENCLUS [4] uses entropy as a criteria to drive the algorithm. However, ENCLUS follows a completely different algorithm to our approach. ENCLUS divides the hyperspace recursively, considering an additional dimension in each iteration. For each subspace, ENCLUS estimates its density and entropy and determines if it satisfies the goodness criteria: its entropy has to be lower than a threshold. The authors of ENCLUS prove the relationship of their entropic measure with the definition of density. However, it is not possible to translate either the algorithm or the relationships to the area of categorical clustering, since the notion of density has no intuitive meaning when the attributes are categorical. The density estimation necessitates of a definition of hypervolume of the subspace, which without setting an arbitrary distance between two values of a categorical attribute (e.g., blue and red) is impossible to calculate.

In the area of clustering categorical records, a few recent publications are worth mentioning. In [16], the authors address the problem of clustering transactions in a market basket database. To do so, each frequent itemset is represented as a hyperedge in a weighted hypergraph. The weight of the graph is computed as the average of the confidences for all possible association rules that can be generated from the itemset. Then, a hypergraph partitioning algorithm is employed to partition the items, minimizing the weight of the cut hyperedges. The results is a clustering of items. However, the algorithm does not produce a clustering of the transactions and it is not obvious how to obtain one from the item clusters. A related paper by Gibson et al [12] also treats categorical clustering as hypergraph partitioning, but uses a less combinatorial approach to solving it, based on non-linear dynamical systems.

CACTUS [10], is an agglomerative algorithm that uses the author's definitions of *support*, *strong connection* and *similarity* to cluster categorical data. Support for an attribute value pair (a_i, a_j), where a_i is in the domain of attribute A_i and a_j in the domain of attribute A_j is defined as the number of tuples that have these two values. The two attributes a_i, a_j are strongly connected if their support exceeds the value expected under the attribute-independence. This concept is then extended to sets of attributes. A cluster is defined as a region of attributes that are pairwise strongly connected, no sub-region has the property, and its support exceeds the expected support under the attribute-independence assumption. Similarity is used to connect attribute values (a_1, a_2) of the same attribute A_i, and it measures how many "neighboring" values x belonging to other attributes exist, such that a_1, x and a_2, x have positive support. Using support and similarity, CACTUS defines inter-attribute and intra-attribute summaries which tell us how related values from different and the same attribute are respectively. CACTUS uses these summaries to compute the so-called cluster projections on individual attributes and use these projections to obtain candidate clusters on a pair

of attributes, extending then to three and more attributes. CACTUS, obviously, bases its clustering decisions on the "neighboring" concept of similarity, similar to the algorithms described in [16, 12].

Hierarchical agglomerative methods (e.g., [27, 19, 18]), cluster points by iteratively merging clusters (at the start every point is a cluster by itself). For instance, the *single linkage* method [27] begins searching for the two most similar points in a similarity matrix and puts them in the same cluster. Then it searches for the most similar point to those in the cluster, making this point part of the cluster. At every step, the algorithm looks for points to join into existing clusters (or among themselves). The problem with agglomerative methods is that they do not scale, since they require the calculation and storage of potentially large similarity matrices. They also lack stability when the data is re-shuffled in the similarity matrix.

ROCK [15] computes distances between records using the Jaccard coefficient (redefined as shown in Section 2.3). Using a threshold, it determines, for each record, who are its neighbors. For a given point p, a point q is a neighbor of p if the Jaccard coefficient $J(p,q)$ exceeds the threshold. Then, it computes the values of a matrix $LINK$, in which the entries $link(p,q)$ are the number of common neighbors between p and q. The algorithm then proceeds to cluster the records in an agglomerative way, trying to maximize for the k clusters (k is a predefined integer) the function $\sum_{i=1}^{k} n_i \sum_{p,q \in C_i}$, where ϕ is the threshold, and $f(\phi)$ is a function selected by the user. The ROCK solution offered in Figure 5 for the data of Figure 4 was found using $\phi = 0.5$ and $f(\phi) = \frac{1-\phi}{1+\phi}$. The choice of $f(\phi)$ is critical in defining the fitness of the clusters formed the the ROCK algorithm, and, as the authors point out, the function is dependent on the data set as well as on the kind of cluster the user is interested in. We feel that choosing the function is a delicate and difficult task for users that may be a roadblock to using ROCK efficiently. In contrast, our algorithm offers a unique, well-defined cluster fitness criteria, that is based on a solid and well understood property: entropy.

4 Our algorithm

Our entropy-based algorithm, COOLCAT, consists of two steps: initialization and incremental step.

4.1 Initialization

The initialization step "bootstraps" the algorithm, finding a suitable set of clusters out of a sample S, taken from the data set ($|s| << N$), where N is the size of the entire data set. We first find the k most "dissimilar" records from the sample set by maximizing the minimum pairwise entropy of the chosen points. We start by finding the two points p_{s_1}, p_{s_2} that maximize $E(p_{s_1}, p_{s_2})$ and placing them in two separate clusters (C_1, C_2), marking the records (this takes $O(|S|^2)$). From there, we proceed incrementally, i.e., to find the record we will put in the j-th cluster, we choose an unmarked point p_{s_j} that maximizes $min_{i=1,..,j-1}(E(p_{s_i}, p_{s_j}))$.

The rest of the sample unmarked points ($|S| - k$), as well as the remaining points (outside the sample), are placed in the clusters using the incremental step.

4.2 Incremental Step

After the initialization, we process the remaining records of the data set (the rest of the sample and points outside the sample) incrementally, finding a suitable cluster for each record. This is done by computing the expected entropy that results of placing the point in each of the clusters and selecting the cluster for which that expected entropy is the minimum. We proceed in the incremental step by bringing a buffer of points to main memory and clustering them one by one. The first batch of points is composed by those records in the sample that were not selected to seed the clusters initially.

1.Given an initial set of clusters $\breve{C} = C_1, \cdots, C_k$

2.Bring points to memory from disk and for each point p do

3. For $i = 1, .., k$

4. Place p in C_i and compute $\bar{E}(\breve{C}^i)$
where $\breve{C}^i$ denotes the clustering obtained by placing p in cluster C_i

5. Let $j = argmin_i(\bar{E}(\breve{C}^i))$

7. Place p in C_j

8. Until all points have been placed in some cluster

Figure 6: Incremental step.

The order of processing points has a definite impact on the quality of the clusters obtained. It is possible that a point that seem a good fit for a cluster at a given point in time, becomes a poor fit as more points are clustered. In order to reduce this effect, we enhanced the heuristic by re-processing a fraction of the points in the batch. After a batch of points is clustered, we select a fraction m of points in the batch that can be considered the worst fit for the clusters they were put in. We proceed to remove these points from their clusters and re-cluster them. The way we figure out how good a fit a point is for the cluster where it landed originally, is by keeping track of the number of occurrences of each of its attributes' values in that cluster. That is, at the end of the batch, we know the values of q_{ij}, for each record i in the batch and each attribute j, where q_{ij} represent the number of times that the value V_{ij} appears in the cluster where i was placed. We convert these numbers into probabilities by dividing q_{ij} by the cluster size (i.e., $\|C_l\|$, where C_l is the cluster where i was placed). Let us call these numbers p_{ij}. For each record, we can compute a fitting probability $p_i = \prod_j(p_{ij}$. Notice that the lower the p_i is, the worst fit the record is in that cluster (we can say that the global combination of attributes is not very common in the cluster). We then sort records according to p_i and select the m records in the batch with lowest p_i as the records to be reprocessed. Each re-processed record is placed in the cluster that minimizes the expected entropy (as done originally in the incremental step).

5 Experimental Results

We conducted a series of experiments to evaluate COOLCAT. The experiments were run in a DELL server equipped with a Pentium III running at 800 MHz, and 1 Gigabyte of main memory, running Red Hat Linux 2.2.14. We used two kinds of data sets: real data sets (for evaluating the quality of our algorithm) and synthetic data sets (for the evaluation of scalability).

5.1 Real data sets

We used the following data sets

- *Archaeological data set*

  Our first data set is a hypothetical collection of human tombs and artifacts from an archaeological site. The data set is taken from [1] and reproduced in Figure 7. The first attribute

Tomb #	Sex	A1	A2	A3	A4	A5	A6	A7	A8
0	M	1	0	0	1	0	0	0	0
1	M	0	0	0	1	0	0	0	0
2	M	1	0	0	1	0	0	1	1
3	F	1	0	1	0	0	0	0	0
4	F	0	0	1	0	0	0	1	0
5	F	1	0	1	0	0	0	1	0
6	M	1	1	0	1	0	0	0	0
7	M	0	1	0	1	1	0	0	0
8	M	1	0	0	1	1	0	0	0
9	M	1	1	0	1	1	0	0	0
10	M	1	1	0	1	1	0	1	1
11	F	0	0	0	0	1	0	0	0
12	F	1	0	0	0	1	0	0	0
13	F	1	0	0	0	1	0	1	0
14	M	1	1	0	1	1	0	0	0
15	F	0	0	0	0	0	1	0	0
16	F	1	0	0	0	0	1	0	0
17	F	0	0	0	0	1	1	0	0
18	F	1	0	0	0	0	0	0	0
19	F	1	0	0	0	1	1	1	1

Figure 7: Archaeological data set.

(not used for clustering but for verification) indicates the sex (M for male, F for female) of the individuals buried. The other eight attributes are binary (1 present, 0 non-present), and represent artifacts types (e.g., ceramics, bracelets, arrow points) that were found (or not found) in the tomb. Although this is not a real data set, it is realistic and we selected it because it provides us with a small enough set to verify and analyze our results.

- *Congressional votes* This data set was obtained from the UCI Machine Learning Repository ([3]) and contains the United States Congressional Voting Records for the year 1984. Each record contains a Congressman's votes on 16 issues. All the attributes are boolean ("yes" or "no"), with a few of the votes containing missing values. We decided to treat missing values as another domain value for the attribute. A classification field with the labels "Democrat," or "Republican" is provided for each record, which are not used for clustering, but can be loosely used for quality measuring. (Some congressmen "crossed" parties to vote.) There are 435 records in the set (267 Democrats and 168 Republicans).

- *Mushroom data set* The mushroom data set was also obtained from the UCI Repository ([3]). Each record describes the physical characteristics (e.g., odor, shape) of a single mushroom. There is a "poisonous," or "edible" field for each mushroom (which is not used for clustering). All of the attributes are categorical and the set contains 8,124 records in all (4,208 edible mushrooms and 3,916 poisonous ones).

5.2 Archaeological Data

Figure 8 show the results of using COOLCAT in the data set of Figure 7. We performed experiments with two different number of clusters, i.e., 3, and 6. We conducted experiments with the original data set (which we label "independent"), and a modified data set in which we grouped attributes in the following way: $(1), (24), (26), (34), (35), (46), (78)$, to reflect the correlations found among the attributes of the set. Moreover, for 3 clusters, we also conducted "brute force" experiments, in which we found the optimum clustering, i.e., that for which the expected entropy was the minimum. We did this to compare how well our heuristic (COOLCAT) performed. We also report in the table the best results found by ROCK (which have to be found by varying the parameter ϕ over a range of values). The results shown in Figure 8 show that the expected entropy function does a good job in

Number of clusters = 3				
Entropy-based	Independent		$\bar{CU}$	Entropy (Sex)
		COOLCAT	1.062121	0.846717
		Brute Force	1.211111	0.000000
Entropy-based	Correlated		$\bar{CU}$	Entropy (Sex)
		COOLCAT	1.211111	0.000000
		Brute Force	1.121111	0.000000
Rock $\phi = 0.75$			0.722917	0.800000
Number of clusters = 6				
Entropy-based	Independent	COOLCAT	2.008333	0.000000
Entropy-based	Correlated	COOLCAT	1.990000	0.000000
Rock $\phi = 0.75$			1.998333	0.000000

Figure 8: Results for the Archaeological data set

Expected entropy	COOLCAT, 3 clusters, m				brute force
	0%	10% (2)	20% (4)	40% (8)	
Independent	4.2984	4.2964	4.3146	4.2790	3.9557
Correlated	4.4494	4.4410	4.4219	4.3930	4.2365

Figure 9: Expected entropy of the solutions on the Archaeological data as m varies.

clustering this data. In all cases, both the $\bar{CU}$ function and the external entropy of the brute force case (optimum) are better than those found for the best ROCK solution. Particularly encouraging is the fact that the external entropy for the variable SEX (which the authors of the data set indicated as the one being more correlated with the clusters), is 0, so a perfect separation is achieved. On the other hand, COOLCAT (our heuristic) results are very similar to the optimum ones, and better than those obtained by ROCK. In the case of 6 clusters (where we simply could not run the brute force approach), COOLCAT obtained an external entropy (with respect to SEX) of 0 in both the correlated and independent cases. COOLCAT's results shown in Figure 8 are obtained without re-processing of points.

Figure 9 shows how the expected entropy of the solution found by COOLCAT decreases as the percentage of points re-processed per batch increases. Since this is a small set, the batch consists of all the records (20), and the number of reprocessed points was 2,4, and 8 ($m = 10\%, 20\%, 40\%$). The results indicate that after a certain percentage, the benefit in re-processing points is small. In the worst case re-processing 20% of the points (for the independent data set) resulted in an expected entropy 9% worse than the optimum solution (4.3146 vs. 3.9557). The results reported on Figure 9 correspond to the means of 500 runs (each run follows a different order of processing of the records). Both COOLCAT and ROCK took 0.01 seconds to find a solution for this data set.

Metric	COOLCAT, m				ROCK
	0%	10% (10)	20% (20)	40% (40)	
$\bar{CU}$	2.928649	2.905248	2.932904	2.924817	2.628234
Entropy (political affiliation)	0.501475	0.525497	0.487478	0.506039	0.499362
Expected entropy	13.9632	13.9585	13.9283	13.9079	-
Running times(sec.)	0.16	0.26	0.28	0.29	0.51

Figure 10: Results for COOLCAT and ROCK in the Congressional Voting data set

Metric	COOLCAT, m				ROCK
	0%	10% (20)	20% (40)	40% (80)	
$\bar{CU}$	7.089928	7.110608	7.090686	7.125781	6.882149
Entropy (edible)	0.023374	0.030934	0.023575	0.03177	0.012389
Expected entropy	9.8744	9.8551	9.9241	10.0216	-
Running times(sec.)	2.91	4.16	4.51	5.90	254.91

Figure 11: Results for COOLCAT and ROCK in the Mushroom data set

5.3 Congressional Voting results

Figure 10 summarizes the results obtained by COOLCAT in the Congressional Voting records (no grouping of attributes was performed), for four values of m. The results obtained for various sample sizes are extremely stable. The $\bar{CU}$ values for the clusterings obtained with COOLCAT are, in all the cases superior to the one obtained by ROCK. (The values show small fluctuations on our results as m changes, while all the values are at least 10% better than ROCK's value.) The external entropy values of all the solutions (COOLCAT's and ROCK's) are comparable, and small enough to indicate a good separation into Democrats and Republicans by the clusterings found. The expected entropy of the solution found by COOLCAT as m varies is reported in the table of Figure 10, showing, as expected, a decrease in entropy as m increases. The buffer size (batch) in this experiment was 100 records, making the number of re-processed points 10,20, and 40 ($m = 10\%, 20\%, 40\%$). The trend stabilizes quickly, indicating that a small amount of re-processing is sufficient. (Again, these numbers correspond to the means of 500 runs.)

5.4 Mushroom results

Figure 11 shows the results of using COOLCAT in the Mushroom data set for different values of m (percentage of reprocessed points per batch). No grouping of attributes was performed. We also tried samples of various sizes with equal results. Again, the $\bar{CU}$ values for all the solutions found by COOLCAT are larger than the one for the ROCK solution. The external entropy values found in all cases are comparable and very small, indicating a good separation of edible and non-edible mushrooms in the clusters. The running times are reported in the table: those for COOLCAT are a fraction (at most 2.3%) of the running time of ROCK. (The reported values for the expected entropy as m increases are again means of 500 runs.) The buffer size in this case was 200 records, so the number of re-processed points was 20, 40, and 80.

5.5 Synthetic data set

We used a synthetic data generator ([5]) to generate data sets with different number of records and attributes. We used these data sets to test the scalability of COOLCAT. The results are shown in the graph of Figure 12, where the y-axis shows the execution time of COOLCAT in seconds, and the x-axis the number of records (in multiples of 10^3), for four different number of attributes ($A = 5, 10, 20, 40$). In all the cases, COOLCAT behaves linearly with respect to the number of records, due to the incremental nature of the algorithm (it processes each record in the data set at most twice: those that are selected for re-processing are clustered twice, the rest only once; moreover, points are brought from disk to memory only once). We used for these experiments an m equal to 20%, and a buffer size of 300 records. Notice that in this experiment, we do not report running times for ROCK. The reason for this is that ROCK is designed to be a main memory algorithm. In [15], the authors make it explicit that ROCK deals with large data sets by using random sampling (not by looking at the entire set). Therefore, it would have been unfair to compare COOLCAT's running times with those of ROCK (over samples of the sets).

6 Conclusions

In this paper we have introduced a new categorical clustering algorithm, COOLCAT, based in the notion of entropy. The algorithm groups points in the data set trying to minimize the expected entropy of the clusters. The experimental evaluation supports our claim that COOLCAT is an efficient algorithm, whose solutions are stable for different samples (and sample sizes) and it is scalable for large data sets (since it incrementally adds points to the initial clusters). We have evaluated our results using category utility function, and the external entropy which determines if the clusters have significance with respect to external variables (i.e., variables not used in the clustering process). In our comparisons with ROCK, COOLCAT always shows a small advantage in terms of the quality measures (CU and external entropy). However, the real advantage of COOLCAT resides in the fact that ROCK is extremely difficult to tune (finding the right ϕ), while COOLCAT's behavior to its only parameter (m) is extremely stable: small values of m are sufficient to obtain a good result. In the largest data set for which we compared both techniques (Mushrooms), COOLCAT had a significantly better running time.

In the future, we plan to use COOLCAT in applications related to document (WEB) clustering and bioinformatics.

7 Acknowledgments

We like to thank Vipin Kumar and Eui-Hong (Sam) Han for lending us their implementation of ROCK.

References

[1] M.S. Aldenderfer and R.K. Blashfield. *Cluster Analysis.* Sage Publications, (Sage University Paper series on Quantitative Applications in the Social Sciences, No. 44), 1984.

[2] D. Barbará and P. Chen. Using the fractal dimension to cluster datasets. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA*, August 2000.

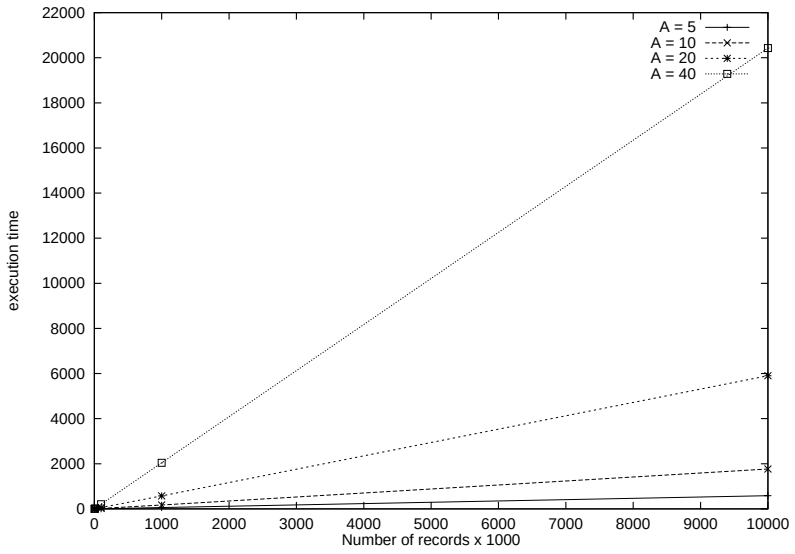

Figure 12: COOLCAT's performance for the synthetic data sets: response time (in seconds) vs. the number of records in the data set (in multiples of 10^3), for different number of attributes ($A = 10, 20, 40$).

[3] C. Blake(Librarian). UCI Machine Learning Repository. http://www.ics.uci.edu/ mlearn/MLRepository.

[4] C. CHen, A.W. Fu, and Y. Zhang. Entropy-based Subspace Clustering for Mining Numerical Data. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA*, August 1999.

[5] DataGen. Data Generator: Perfect data for an imperfect world. http://www.datasetgenerator.com/.

[6] S. Dolnicar, F. Leisch, A. Weingessel, C. Buchta, and E. Dimitriadou. A comparison of several cluster algorithms on artificial binary data scenarios from travel market segmentation. Working Paper 7, SFB 'Adaptive Information Systems and Modeling in Economics and Management Science', http://www.wu-wien.ac.at/am, April 1998.

[7] R.O. Duda and P.E. Hard. *Pattern Classification and Scene Analysis.* Wiley-Interscience, New York, 1973.

[8] M. Ester, H.P. Kriegel, and X. Wu. A database interface for clustering in large spatial databases. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining, Montreal, Canada*, August 1995.

[9] M. Ester, H.P. Kriegel, and X. Wu. A density-based algorithm for discovering clusters in large spatial database with noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining, Portland, Oregon*, August 1996.

[10] V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS-Clustering Categorical Data Using Summaries. In *Proceedings of the ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA*, 1999.

[11] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.

[12] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering Categorical Data: An Approach Based on Dynamical Systems. In *Proceedings of the International Conference on Very Large Databases (VLDB), New York, NY*, September 1998.

[13] A. Gluck and J. Corter. Information, uncertainty, and the utility of categories. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, 1985.

[14] S. Guha, R. Rastogi, and K. Shim. CURE: A clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data, Seattle, WA*, May 1998.

[15] S. Guha, R. Rastogi, and K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. In *Proceedings of the 15th International Conference on Data Engineering, Sydney, Australia*, April 1999.

[16] E.H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering based on association rule hypergraphs. In *Proceedings of the SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, June 1997.

[17] A.K. Jain and R.C. Dubes. *Algorithms for clustering data.* Prentice Hall, 1988.

[18] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons, New York, 1990.

[19] L.L. McQuitty. Hierarchical linkage analysis for the isolation of types. *Ed. Phychol. Measurement*, 20:56–67, 1960.

[20] T.M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[21] R.T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the VLDB Conference, Montreal, Canada*, 1997.

[22] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 1983.

[23] J. Rissanen. *Stochastic complexity in statistical inquiry.* World Scientific Pub., 1989.

[24] C.E. Shannon. A mathematical theory of communication. *Bell System Techical Journal*, pages 379–423, 1948.

[25] D. Slaven. The page of entropy. http://www.svsu.edu/ slaven/Entropy.html.

[26] P. Sneath and R. Sokal. *Numerical Taxonomy.* W. H. Freeman, San Francisco, 1973.

[27] P.H.A. Sneath. Some thoughts on bacterial classifications. *Journal of General Microbiology*, 17:201–226, 1957.

[28] I.H. Witten and E. Frank. *Data Mining.* Morgan Kaufmann, 2000.

[29] R. Zhang, R. Ramakrishnan, and M.Livny. Birch: An efficient data clustering method for very large databases. In *Proceedingfs of the ACM SIGMOD Conference on Data Management, Montreal, Canada*, June 1996.