Finding Representative Set from Massive Data

Feng Pan, Wei Wang
University of North Carolina
at Chapel Hill
panfeng@cs.unc.edu
weiwang@cs.unc.edu

Anthony K. H. Tung
National University
of Singapore
atung@comp.nus.edu.sg

Jiong Yang
Case Western Reserve
University
jiong@eecs.cwru.edu

Abstract

In the information age, data is pervasive. In some applications, data explosion is a significant phenomenon. The massive data volume poses challenges to both human users and computers. In this project, we propose a new model for identifying representative set from a large database. A representative set is a special subset of the original dataset, which has three main characteristics: It is significantly smaller in size compared to the original dataset. It captures the most information from the original dataset compared to other subsets of the same size. It has low redundancy among the representatives it contains. We use information-theoretic measures such as mutual information and relative entropy to measure the representativeness of the representative set. We first design a greedy algorithm and then present a heuristic algorithm that delivers much better performance. We run experiments on two real datasets and evaluate the effectiveness of our representative set in terms of coverage and accuracy. The experiments show that our representative set attains expected characteristics and captures information more efficiently.

1. Introduction

Given a huge dataset, generating an accurate synopsis are of great significance to both people and computer and are used in many applications. In this project, we propose a model for identifying representative set, which is a subset of the original dataset, as a form of synopsis. A representative set consists of selective samples from the original dataset. It captures significant information in the original dataset more efficiently than any random samples can. Applications of representative set include but are not limited to the following two.

- In unsupervised clustering, user interference is restricted to setting a few global parameters such as the number of clusters. Recent research has suggested that moderate user interaction will greatly enhance the interpretability and usefulness of the clustering result. The representative set offers an opportunity to allow sensible user interference. Since a representative set is significantly smaller than the original dataset, specialists can observe every representative thoroughly and decide on its class label or its relationships with others. And these information are valuable references to improve the quality of the final clustering results.

- A Google search can easily generate thousands of entries that satisfy the search criterion. It is infeasible for a user to examine every entry. However, if we can generate a representative set of the entries and each representative can stand for a typical category or topic, the user can quickly comprehend the scope of the search results and determine ways to focus the search. Further, these representatives are also good reference coordinates for meaningful document clustering.

A good representative set should capture the most information from the original dataset compared to other subsets of the same size. Also, it should have low redundancy. Algorithms such as Maximum Coverage [9] can generate a subset that captures original information from a dataset, but may only work well in a balanced dataset, where the number of transactions from each class is similar. However, the maximum coverage approach does not generate good representative sets that take into consideration low redundancy. Good performance of the maximum coverage approach depends on an appropriate choice of similarity function and similarity threshold.

General cluster algorithms address the problem to some extent, especially representative-based clustering algorithms such as the k-medoid clustering [8]. However, as we will show in the experiment section, generating a representative set in advance can help the processing of representative-based clustering algorithms.

We use information-theoretic measures such as mutual information and relative entropy [3, 5] to search for good representative sets. To meet the expectation that the representative set should capture the most information and avoid redundancy, we design an objective function and a greedy algorithm to make the optimal choice at each step when selecting a new representative. We also design a simplified version of the greedy algorithm which employs heuristics to achieve much better performance.

The rest of this paper is organized as follows: We define the key terms in Section 2. The detail of our algorithms is in Section 3. We report the experiment results in Section 4. The related work is in Section 5. We conclude in Section 6.

2. Preliminary

We present some information-theoretic measurements in this section. Since we have two requirements for a good representative: high coverage and low redundancy, we employ two information-theoretic measurements. We use mutual information to measure the coverage of the representatives; good representatives that capture most information in the original dataset should have a large mutual information value with respect to the features of the dataset. We will use relative entropy to measure the redundancy between the representatives. A high relative entropy between representatives infers a low redundancy. Therefore, as we will see in the next section, our objective function will consist of two parts which are equally important. We will define terms and provide examples related to mutual relative entropy and mutual information in this section. The following table contains the list of notations we will use in this section.

Table 1. Notations

E	the entire element set, $E = \{e_1, e_2 ... e_n\}$
e, e_i	single element, $e, e_i \in E$
E_i	subset of E, $E_i \subseteq E$
F	the entire feature set, $F = \{f_1, f_2 ... f_m\}$
f, f_i	single feature, $f, f_i \in F$
R	the representative set, $R \subseteq E$
r, r_i	single representative, $r, r_i \in R$
$L(r_i)$	set of elements related to representative r_i $L(r_i) \subseteq E$
E_Θ	set of elements not related to any representative in R, $E_\Theta \subseteq E$
r_Θ	the virtual representative for E_Θ, $L(r_\Theta) = E_\Theta$
T	random variable over domain of E
A	random variable over domain of F
M	random variable over domain of $R \cup \{r_\Theta\}$

The datasets studied in the paper are a set of elements in a given feature space. Each element is described by a set of features. For each feature, a binary variable is used to depict the presence or absence of the feature. We represent the entire element set as E and the entire feature set as F. We transform the original dataset into a two-dimensional binary table whose rows correspond to elements and columns correspond to features in the original dataset. For example, Figure 1(a) is the original dataset which contains five elements and five distinct feature values. Figure 1(b) is the transformed dataset, e_1 has value 1 in column f_1 because e_1 has feature f_1 in the original dataset.

	features
e_1	$f_1\ f_2\ f_3\ f_4$
e_2	$f_2\ f_3\ f_4$
e_3	$f_1\ f_4\ f_5$
e_4	$f_3\ f_4\ f_5$
e_5	$f_1\ f_5$

(a) Original Dataset

	f_1	f_2	f_3	f_4	f_5
e_1	1	1	1	1	0
e_2	0	1	1	1	0
e_3	1	0	0	1	1
e_4	0	0	1	1	1
e_5	1	0	0	0	1

(b) Transformed Table

Figure 1. Example Data

By normalizing each row in Figure 1(b), we can view an element as a distribution in the feature domain. Table 2 shows the distribution after normalizing the data in Figure 1(b).

Table 2. Distribution Table

	f_1	f_2	f_3	f_4	f_5
e_1	0.25	0.25	0.25	0.25	0
e_2	0	0.33	0.33	0.33	0
e_3	0.33	0	0	0.33	0.33
e_4	0	0	0.33	0.33	0.33
e_5	0.5	0	0	0	0.5

We define two random variables, T and A, in the element domain and feature domain respectively. Giving equal weight to each element $e_i \in E$, we define:

$$p(T = e_i) = \frac{1}{|E|}, e_i \in E$$

According to the distribution table, we obtain the conditional probability $P(A|T)$. For example, $P(A = f_1|T = e_1) = 0.25$. For convenience, we use $P(f_1|e_1)$ to represent $P(A = f_1|T = e_1)$. For each subset E_i of E, we define the probability distribution function in the following way:

$$P(E_i) = \frac{|E_i|}{|E|}, E_i \subseteq E$$

$$P(A|E_i) = \sum_{e \in E_i} \frac{P(e)}{P(E_i)} P(A|e) = \frac{1}{|E_i|} \sum_{e \in E_i} P(A|e)$$

For two element subsets, the relative entropy can be used to measure the difference. It is defined on the two corresponding probability distributions.

Definition 2.1 ***Relative entropy between element sets*** *Given two element sets E_i and $E_j \subseteq E$, the relative entropy between them is the relative entropy or* Kullback-Leibler *divergence between their distributions in the feature domain:*

$D_{KL}[E_i||E_j] = D_{KL}[P(A|E_i)||P(A|E_j)]$

$= \sum_{f \in F} P(f|E_i) log \frac{P(f|E_i)}{P(f|E_j)}$

To avoid the problem of indefinite value when $P(f|E_i)$ or $P(f|E_j)$ equals 0, we will use a real number close to 0 to replace 0 in implementation such as 10^{-10}. In the discussion below, we will also use the relative entropy between two elements where single element is considered as a degeneracy of element set.

A representative is a typical element of the set E. Our algorithm aims to find a small representative set R from a huge collection of elements. We give a general definition of the representative set as follows:

Definition 2.2 ***Elements related to Representative*** *Given a representative r, an element e is* **related** *to r if $D_{KL}(r||e) < (min_{e_i \in E-\{r\}} D_{KL}(r||e_i)) * t_{max}$, where $t_{max} \geq 1$, that is, the relative entropy between r and e is within a certain range of the minimal relative entropy between r and all other element in E. t_{max} is a parameter used to control the range. We use $L(r)$ to denote the set of elements related to r. When $t_{max} < 1$, $L(r) = \{r\}$.*

In principle, an element may be related to several representatives which will make the problem complicated and make trouble on the random variable M which we will define later. Therefore, we make some modification on Definition 2.2 to resolve this issue. We generate representatives one by one, so that when we pick elements related to a new representative, we only consider those elements that are not related to any previously chosen representatives. By doing so, each element will be related to at most one representative. Similar approach was used in some max coverage approaches.

Definition 2.3 ***Representative Set*** *A representative set R is a subset of E. For each representative $r_i \in R$, we can get its related element set $L(r_i)$, $L(r_i) \subseteq E$. Given a representative set $R = \{r_1, r_2, ..r_n\}$, $E = L(r_1) \cup L(r_2) \cup .. \cup L(r_n) \cup E_\theta$. E_θ contains all the elements which are not related to any representative in R.*

In Definition 2.3, E_θ contains all the elements not related to any representative. For convenience in explaining our algorithm, we consider E_θ the related set of a special representative r_θ that does not exist in the dataset, $L(r_\theta) = E_\theta$.

We define a random variable M over the representative set and r_θ. Given a representative set $R = \{r_1, r_2, ..r_n\}$,

$$P(M = r_i) = \frac{|L(r_i)|}{|E|}, P(M = r_\theta) = \frac{|E_\theta|}{|E|}$$

$$P(A|M = r_i) = \frac{1}{|L(r_i)|} \sum_{e \in L(r_i)} P(A|e)$$

$$P(A|M = r_\theta) = \frac{1}{|E_\theta|} \sum_{e \in E_\theta} P(A|e)$$

For convenience, we will use $P(f_1|r_1)$ to represent $P(A = f_1|M = r_1)$ later.

Mutual information is a measure of the relationship between two random variables. We can use mutual information between random variables M and A, $I(M, A) = H(A) - H(A|M)$, to measure the information captured when representing the original dataset with the representative set R. Intuitively, $I(M, A)$ measures how much variation in the feature domain A is captured by a representative set. The higher, the better. Given two representative sets R_1 and R_2, $R_1 = \{r_1, r_2, .., r_n\}$, $R_2 = R_1 \cup \{r_{n+1}\}$, and their corresponding $E_{1\theta} = L(r_{1\theta})$ and $E_{2\theta} = L(r_{2\theta})$, we get $L(r_{2\theta}) = L(r_{1\theta}) - L(r_{n+1})$. Using this equality, we can calculate the difference between $I(M_1, A)$ and $I(M_2, A)$:

$\Delta I(M_2, M_1) = I(M_2, A) - I(M_1, A)$

$= H(A|M_1) - H(A|M_2)$
$= \frac{|L(r_{2\theta})|}{|E|} D_{KL}[p(A|r_{2\theta})||p(A|r_{1\theta})] +$
$\frac{|L(r_{n+1})|}{|E|} D_{KL}[p(A|r_{n+1})||p(A|r_{1\theta})]$

Since relative entropy is always positive, we know that R_2 retains more information than R_1.

Property 2.1 (Monotonicity) *Given a representative set R, if we generate a new representative set R' by adding a new representative to R, we can always have $I(M', A) \geq I(M, A)$. M is the random variable defined over R and $\{r_\theta\}$. M' is the random variable defined over R' and $\{r'_\theta\}$.*

2.1. Objective Function & Problem Definition

Property 2.1 suggests that we may use a greedy algorithm to successively pick representatives that offer the highest mutual information. Starting from an empty set $R = \emptyset$, each time we add a new representative to R which can increase the mutual information most – meaning it can capture more original information than any of the remaining non-representative elements. At the same time, we should also minimize the redundancy between the new representative and existing representatives. We measure the redundancy between two representatives by their relative entropy. High relative entropy infers big difference between the probability distribution of the two representatives and thereby small redundancy. Combining these two factors, we define our objective function as follows:

$$f(r_{new}, R) = \Delta I(M_{new}, M) + min_{r \in R}(D_{KL}(r_{new}||r))$$

The formal definition of our problem is as follows.

Problem Definition: Given a dataset which consists of elements $E = \{e_1, e_2, ..., e_n\}$, and an empty representative set R, add k representatives into R one by one such that at each step, the objective function $f(r_i, R)$ can be maximized.

3. Algorithms

In this section, we will first describe the greedy algorithm we used to generate the representative set. And then, we will give a simplified version of the greedy algorithm.

3.1. Greedy Algorithm

We use a greedy algorithm to select new representatives at each step which can maximize the objective function f until we have generated the required number of representatives. A formal description of the greedy algorithm is given in Algorithm 1. As we can see, the greedy algorithm is simple and easy to implement.

Algorithm 1 *Greedy Algorithm: Representative Set*

Input: Dataset which can be viewed as element set $E = \{e_1, e_2, ...e_n\}$, Size of representative set, m.
Output: Representative Set R

$R = \{\}$
while $|R| < m$ **do**
  for all $e_i \in E_\theta$ **do**
    calculate $f(e_i, R)$
  $R = R \cup \{e\}$ if $f(e, R) = max_{e_i \in E_\theta}(f(e_i, R))$
  update E_θ

3.2. Simplified Algorithm

The greedy algorithm in the previous section has a computational complexity of $O(m|E|)$, where m is the number of representatives, and $|E|$ is the size of the element set. When the dataset grows, it becomes time-consuming to generate the representative set. In applications in which response time is crucial, such as web search, we need to generate the representative set much faster.

As we look through Algorithm 1, we can find that the cause for the complexity is that at each iteration, we consider each remaining element in the set as a candidate for the next representative. So if we can narrow the candidate set, we can expect a faster performance. According to our objective function in Section 2.1, a good representative maximizes information gain and dissimilarity with other representatives. While it may be difficult to estimate information gain in advance, it is easy to find elements dissimilar to the representatives already found. Since we have calculated the relative entropy between each pair of elements as part of preprocessing, we can use those results to find a set of elements which are most dissimilar to each representative very quickly. We can build the candidate set by taking the union of these dissimilar sets. Similar to Definition 2.2, we can define a dissimilar set for each representative.

Algorithm 2 *Simplified Version of Greedy Algorithm*

Input: Dataset which can be viewed as element set $E = \{e_1, e_2, ...e_n\}$, Size of representative set, m.
Output: Representative Set R

$R = \{\}$
$Candidate = \{\}$
while $|R| < m$ **do**
  for each $r_j \in R$ **do**
    $Candidate = Candidate \bigcup D(r_i)$
  $Candidate = Candidate - \bigcup_{r_i \in R} L(r_i)$
  for all $e_i \in Candidate$ **do**
    calculate $f(e_i, R)$
  $R = R \cup \{e\}$ if $f(e, R) = max_{e_i \in Candidate}(f(e_i, R))$

Definition 3.1 ***Dissimilar set of a representative***
An element e belongs to the dissimilar set of a representative r if and only if $D_{KL}(r||e) > (max_{e_i \in E-\{r\}} D_{KL}(r||e_i)) * t_{min}, t_{min} < 1$. We denote the dissimilar set of representative r as $D(r)$. Parameter t_{min} is used to control the size of the dissimilar set. A smaller t_{min} will result in a larger dissimilar set for a representative.

Definition 3.2 ***Candidate set for next representative***
Given a set of generated representatives $R = \{r_1, r_2, \ldots, r_k\}$, the candidate set for the next representative is: $Candidate = \cup_{i=1..k} D(r_i)$

In fact, the candidate set can be defined in a more general way. With a pre-defined integer x, the candidate set consists of elements which are contained in at least x dissimilar set of representatives. If $x = 1$, candidate set is the union of the dissimilar sets as we defined in Definition 3.2. And if x is the number of representatives, the candidate set is the intersection of the dissimilar sets. In our algorithm, we will use $x = 1$.

The simplified greedy algorithm is described in detail in Algorithm 2. We can expect the algorithm to be faster and less optimal when the parameter t_{min} increases. As we will see in the experiment section, $t_{min} = 0.9$ is a proper value.

4. Applications and Experiments

In this section, we verify the effectiveness of the representative set. We apply the algorithm to different kinds of real-life datasets including the Mushroom dataset and the 20 Newsgroup dataset. All the experiments are conducted

on a PC with PIV 1.6G CPU, 512M main memory and 30G hard drive. The algorithms are implemented in C.

Algorithms

We compare the performance of our algorithm against two others, **MaxCover** and **RandomPick**.

MaxCover is a greedy approach for Maximum k-coverage in [9]. This approach assumes that every element in the dataset has a coverage set which consists of elements similar to it. In our implementation, we define the coverage set of an element in the same way as Definition 2.2.

Definition 4.1 ***Coverage set of element***
The coverage set of element e, C(e), is defined as $C(e) = \{e_i | D_{KL}(e||e_i) < (min_{e_i \in E-\{e\}} D_{KL}(e||e_i)) * c_{max}\}$, $c_{max} \geq 1$.
c_{max} *is a similarity threshold that is analogous to* t_{max}.

In the RandomPick method, we randomly pick a subset of the dataset as representatives. The average performance of 10 runs is reported for each experiment.

Measurements

We use two measurements in experiments: *coverage* and *accuracy*. Coverage measures the percentage of classes that are covered by the representative set. A class is covered if and only if at least one of the representative belongs to that class. Let C(R) be the distinct number of class labels covered by representative set R and $|C|$ be the total number of classes in the dataset, then *coverage* is defined as:

$$coverage = \frac{C(R)}{|C|}$$

Besides the *coverage* measurement, we want to design a more rigid task to show the effectiveness of our representative set. Therefore we design a clustering algorithm using the representative set. Given a representative set, we obtain the class label of each representative[1]. Each remaining element is assigned to the class of its closest representative. The description of the algorithm follows:

Algorithm 3 *Cluster Based on Representative Set*

Input: Dataset which can be viewed as element set $E = \{e_1, e_2, ...e_n\}$
Output: Clustering of the dataset

1: Generate representative set R, $|R| = m$, $m << |E|$
2: retrieve label for each representative in R
3: **for** all $e \in E$ **do**
4: calculate $D_{KL}(r_i||e)$, for $\forall r_i \in R$
5: assign e to representative r if $D_{KL}(r||e) = min_{r_i \in R}(D_{KL}(r_i||e))$

We argue here that a good set of representatives would have the same class label as those elements that are being covered by them. Let $C(E)$ be the number of elements that have the same class label as their nearest representative. Then *clustering accuracy* is given in the form of:

$$clustering\ accuracy = \frac{C(E)}{|E|}$$

For convenience, we will denote this measurement as *accuracy* in later discussions.

4.1. Mushroom Dataset

We use the Mushroom dataset from UCI machine learning archive. It contains 8124 elements and 22 categorical attributes. The elements are in two classes.

We vary the number of representatives from 2 to 10 and compare the coverage. We set the similarity threshold t_{max} and c_{max} to 4. The result is in Figure 2(a). Both our algorithm and MaxCover cover the two classes when enough representatives are generated. However, our algorithm does it faster than MaxCover and RandomPick.

$\|R\|$	Representative Set	MaxCover	RandomPick
2	100%	50%	70%
3	100%	100%	95%
4	100%	100%	90%
5	100%	100%	100%
10	100%	100%	100%

(a) Representative Coverage

$\|R\|$	Representative Set	MaxCover	RandomPick
2	67.9%	51.7%	48.3%
4	75.1%	71.0%	63.5%
8	89.0%	89.2%	79.3%
20	96.3%	96.4%	90.7%
30	100%	96.3%	93.7%

(b) Clustering Accuracy

Figure 2. Mushroom dataset, $t_{max} = 4$**,** $c_{max} = 4$

We also compare the clustering accuracy achieved by the three methods in Figure 2(b). As we can see, the representative method gives the best performance. MaxCover is better than RandomPick, however, since it does not consider the redundancy of the elements selected, it still performs worse than the representative set method.

Though MaxCover and the representative set method are comparable in terms of *coverage* and *accuracy*, the reliable performance of MaxCover depends on a well-defined similarity threshold while the representative set method is much less sensitive to it. Small adjustment of c_{max} may result in poor performance, as shown in Figure 3(a) and 3(b). MaxCover fails to pick any elements from the second class until the 10^{th} representative and gets poor *accuracy*.

[1] Both datasets in our experiments have class labels.

$\|R\|$	Representative Set	Max Cover
2	100%	50%
5	100%	50%
10	100%	100%

(a) Representative Coverage

$\|R\|$	Representative Set	Max Cover
8	89.0%	51.8%
20	98.5%	86.5%
30	100%	89.3%

(b) Clustering Accuracy

Figure 3. Mushroom dataset, $t_{max} = 3$,$c_{max} = 3$

4.1.1 Comparisons with other clustering algorithms

Several other algorithms have been applied on the Mushroom dataset. One of them is the SUMMARY algorithm [11]. This method summarizes the dataset by clustering it into several groups. When SUMMARY has 30 clusters generated, it achieves accuracy of 99.6%. And it does not get 100% accuracy until more than 400 clusters are generated. As we can see in Table 3, our representative set method can capture the information of the original dataset more efficiently and quicker than SUMMARY can.

Table 3. Clustering Accuracy on Mushroom Dataset, Compared with SUMMARY, $t_{max=4}$

Representative Set		SUMMARY	
$\|R\|$	accuracy	♯ clusters	accuracy
30	100%	30	99.6%
50	100%	140	99.93%
...	...	298	99.99%
...	...	438	100%

In comparison with unsupervised clustering methods such as LIMBO [1], the representative set method also performs better. In [1], the reported accuracy on the Mushroom dataset is about 91%. While in our representative set method, specialists only need to check around 30 elements among 8000 elements to achieve the perfect result. The cost of manual processing is small relative to the improvement in accuracy.

4.1.2 Comparison of the greedy algorithm with its simplified version

In this subsection, we compare the performance of the greedy algorithm with that of its simplified version on the Mushroom dataset. Note that the greedy algorithm is a special case when parameter t_{min} is set to 0 in the simplified version. Therefore, we denote the greedy algorithm as a simplified version with $t_{min} = 0$ in this section.

First, we compare their runtime on the dataset. The preprocessing takes about 290 seconds and we exclude that from the figure below since the results are repeatedly used in different runs. As we can see in Figure 4, the simplified algorithm offers bigger performance improvement as t_{min} increases. The two curves of $t_{min} = 0$ and $t_{min} = 0.85$ are close to each other and exhibit similar trend while the curve of $t_{min} = 0.9$ is far below them. The curve of $t_{min} = 0.9$ even converges to a constant value after 40 representatives are identified when $t_{max} = 4$. This can be explained by looking into the number of candidates generated in each iteration. In Figure 5, we plot the number of candidates in each iteration. The curves of $t_{min} = 0$ and $t_{min} = 0.85$ are always close to each other while that of $t_{min} = 0.9$ is lower. On curve $t_{min} = 0.9$ when $t_{max} = 4$, the number of candidates drops dramatically in the last several iterations, which brings down the slope of the runtime growth and makes it logarithmic in Figure 4.

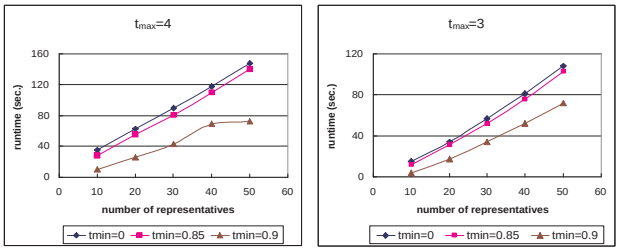


Figure 4. Runtime of different t_{min}

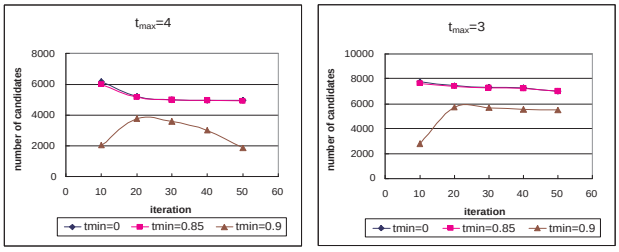


Figure 5. Number of candidates of different t_{min}

Besides runtime, we also compare the accuracy of the clustering algorithm based on the representative sets generated under different t_{min} values. As we can see from Figure 6, when $t_{min} = 0.85$, the performance is the same as $t_{min} = 0$ while at $t_{min} = 0.9$, the performance degrades slightly but is still much better than MaxCov, SUMMARY and LIMBO. This result confirms our discussion in Section 2.

4.2. 20 Newsgroup Dataset

The 20 Newsgroup dataset is a document-words dataset. It consists of 20,000 newsgroup articles from 20 different newsgroups. Since there are more than 30,000 distinct words in all the articles, we conduct a scoring processing which is mentioned in [10]. The top 2000 words with the highest score are selected as features.

We use three subsets of the entire 20 Newsgroup dataset to test our algorithm. Two of the subsets contain articles from two and three newsgroups respectively. Since we get

$\|R\|$	$t_{min} = 0$	$t_{min} = 0.85$	$t_{min} = 0.9$
10	89.2%	89.2%	79.6%
20	96.3%	96.3%	96.3%
30	100%	100%	98.9%
40	100%	100%	99.9%
50	100%	100%	99.9%

(a) Clustering Accuracy, $t_{max} = 4$

$\|R\|$	$t_{min} = 0$	$t_{min} = 0.85$	$t_{min} = 0.9$
10	89.2%	89.1%	79.1%
20	98.5%	98.5%	98.3%
30	100%	100%	98.8%
40	100%	100%	99.9%
50	100%	100%	99.9%

(b) Clustering Accuracy, $t_{max} = 3$

Figure 6. Different t_{min} on Mushroom Dataset

similar results as the Mushroom dataset on these two subsets, we won't present the detailed results of them in this paper. Interested readers can refer to technical report [7].

The third subset is the mini 20 newsgroup dataset which is a reduced version of the full 20 newsgroup dataset. It consists of the same set of 20 newsgroup topics, but each topic contains only 100 articles. We want to test the performance of the three algorithms with respect to the complexity of the data. In this case, the number of newsgroups included in the dataset is a good indicator of the data complexity. Because of the different characteristics of the elements in this mini 20 newsgroup dataset, we will set $t_{max} < 1$ and $c_{max} = 1.1$ in all the following experiments in this section.

First, we compare the methods on the mini 20 newsgroup data. The results are in Figure 7(a) and 7(b). As we can see, in both accuracy and coverage, our representative set method outperforms the other two methods.

	Coverage		
$\|R\|$	Representative Set	MaxCover	RandomPick
20	70%	55%	65%
40	85%	80%	88.5%
60	100%	90%	92%
80	100%	95%	100%
100	100%	100%	99%

(a) Representative Coverage

	Accuracy		
$\|R\|$	Representative Set	MaxCover	RandomPick
20	23.8%	12.5%	18.3%
40	32.5%	21.2%	21.7%
60	37.5%	26.1%	27.2%
80	38.8%	30.3%	28.8%
100	41.6%	32.6%	29.0%

(b) Clustering Accuracy

Figure 7. Mini 20 newsgroup, $t_{max} < 1$, $c_{max} = 1.1$

In order to show the change of performance by dataset containing different number of topics(classes), we start with a subset of the mini 20 Newsgroup consisting of 2 topics and add two topics into the dataset each time until it includes all 20 topics. For each of these dataset, we generate 60 representatives to study the accuracy and coverage. The changes of performance are shown in Figure 8.

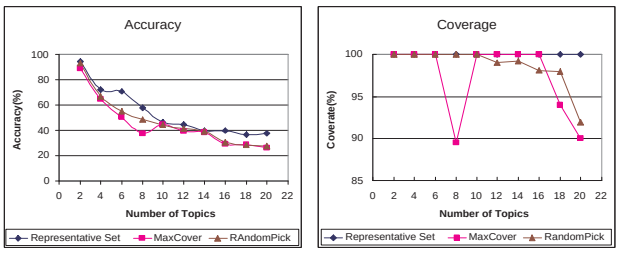


Figure 8. Performance of 60 representatives

All three methods exhibit degrade *accuracy* when more topics are added into the dataset. However our algorithm is always better than the other two. The *accuracy* of MaxCover and RandomPick get close when number of topics is large because each elements is similar to a set of other elements and the size of the similar set has a small deviation when there are large number of topics in the dataset. The choice made by MaxCover is then close to random.

For *coverage*, our algorithm maintains the same performance while other two methods fail to cover all topics when the number of topics increases. The MaxCover method has a big drop on coverage when 8 topics are included. This is because of the characteristic of the 8-topic subsets, i.e., several similar topics are included. And while 2 more topics are added in, the characteristic of the new subset changes.

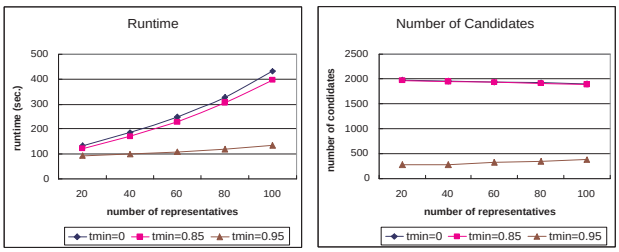
4.2.1 Comparison of the greedy algorithm and its simplified version

As in the previous section, we will compare the runtime performance of our algorithm by varying parameter t_{min}.

We set t_{min} to 0, 0.85 and 0.95 to show its effects. As we can see in Figure 9(a), when we set t_{min} to 0.95, runtime drops dramatically. That is because when $t_{min} = 0.95$, the size of the candidate set for each iteration is small, which can been seen in Figure 9(b).

Besides runtime, we also compare the goodness of the representative sets generated under different t_{min} by the clustering algorithm. We present the accuracy in Table 4.

When t_{min} is set to 0.85, the simplified algorithm achieves the same accuracy as the greedy algorithm. And when t_{min} is set to 0.95, its accuracy is slightly worse than the greedy algorithm, however, the results are still better than that of the MaxCov and RandomPick methods as in Figure 7(b). The slight degrade in accuracy brings the significant improvement in runtime as shown in Figure 9(a).



(a) Runtime (b) Number of Candidates

Figure 9. Performance of different t_{min}

Table 4. Accuracy of different t_{min}

$\|R\|$	$t_{min} = 0$	$t_{min} = 0.85$	$t_{min} = 0.95$
20	23.8%	23.8%	23.1%
40	32.5%	32.5%	31.5%
60	37.5%	37.5%	33.3%
80	38.8%	38.8%	34.5%
100	41.6%	41.6%	38.2%

In all the experiments above, our representative set method always outperforms MaxCover and RandomPick. This shows the effectiveness of our representative sets.

5. Related Work

LIMBO[1] is an hierarchical clustering algorithm based on Information Bottleneck framework. It produces a compact summary model of the data in the first and then employs Agglomerative Information Bottleneck(AIB) algorithm to work on the summarized data. By summarizing the data, LIMBO can handle larger dataset than AIB can.

In [10], a two-phase clustering algorithm is designed for document clustering. The algorithm first performs clustering on words, and then on documents, using the generated word clusters. Its runtime complexity is around $O(mn^2)$, where m is the number of required clusters and n is the size of dataset. While our method takes only $O(mn)$.

Storyline [6] is an approach for clustering web pages using graphic theorem. It builds a bipartite document-term graph and figures out each dense sub-bipartite graph which is actually a set of closely related pages and terms and can be summarized into a cluster. One problem with this method is that though it can cluster web pages into groups, it may not find a proper representative for each group.

Max Coverage [9] can handle the problem we studied in this paper by selecting elements which are similar to most of the elements in the dataset. However Max Coverage cannot capture original information as much as the our method since it only considers coverage while omitting redundancy.

In [2], a semi-supervised clustering method based on information theory performs clustering using predefined constraints. However, to get better performance, the algorithm tends to require more constraints which may be difficult to generate manually.

In [4], a word clustering algorithm replaces the classical feature selection method on document-words datasets. In [4], words are clustered in a supervised way. Instead of using mutual information between words and documents, it maintains mutual information between words and classes.

6. Conclusion

In this paper, we have defined a special subset — the representative set — of the dataset. A representative set is a small subset of the original dataset, captures most original information compared to other subsets of the same size and has a low redundancy. We first design a greedy algorithm to generate the representative set. Then we build a simplified version based on the greedy algorithm for faster and better performance. Our experiments show that the representative set attains the desired characteristics and captures information more efficiently than other methods.

Acknowledgement

This research was partially supported through NSF grant IIS-0448392.

References

[1] P. Andritsos, P. Tsaparas, R. J. Miller, and K. C. Sevcik. Limbo: Scalable clustering of categorical data. *Hellenic Database Symposium*, 2003.

[2] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. *KDD'04 Proceedings*, 2004.

[3] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, New York, 1991.

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

[5] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, New York, 2001.

[6] R. Kumar, U. Mahadevan, and D.Sivakumar. A graph-theoretic approach to extract storylines from search results. *KDD'04 Proceedings*, 2004.

[7] F. Pan, W. Wang, A. K. H. Tung, and J. Yang. Finding representative set from massive data. *Technical Report: TR05-014*, 2005.

[8] S. V. R. Kannan and A. Veta. On clusterings: good, bad, and spectral. In *IEEE Annual Symposium on Foundations of Computer Science*, 2000.

[9] D. S.Hochbaum and A. Pathria. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Quarterly*, (45):615–627, 1998.

[10] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. *ACM SIGIR 2000*, 2000.

[11] J. Wang and G. Karypis. Summary: Efficiently summarizing transactions for clustering. *ICDM'04 Proceedings*, 2004.