Efficiently Clustering Transactional Data with Weighted Coverage Density

Hua Yan
Computational Intelligence Laboratory
School of Computer Science and Engineering
University of Electronic Science & Tech. of China
Chengdu, 610054 P.R.China
huayan@uestc.edu.cn

Keke Chen
College of Computing
Georgia Institute of Technology
Atlanta, GA30280 USA
kekechen@cc.gatech.edu

Ling Liu
College of Computing
Georgia Institute of Technology
Atlanta, GA30280 USA
lingliu@cc.gatech.edu

ABSTRACT

In this paper, we propose a fast, memory-efficient, and scalable clustering algorithm for analyzing transactional data. Our approach has three unique features. First, we use the concept of Weighted Coverage Density as a categorical similarity measure for efficient clustering of transactional datasets. The concept of weighted coverage density is intuitive and allows the weight of each item in a cluster to be changed dynamically according to the occurrences of items. Second, we develop two transactional data clustering specific evaluation metrics based on the concept of large transactional items and the coverage density respectively. Third, we implement the weighted coverage density clustering algorithm and the two clustering validation metrics using a fully automated transactional clustering framework, called SCALE (Sampling, Clustering structure Assessment, cLustering and domain-specific Evaluation). The SCALE framework is designed to combine the weighted coverage density measure for clustering over a sample dataset with self-configuring methods that can automatically tune the two important parameters of the clustering algorithms: (1) the candidates of the best number K of clusters; and (2) the application of two domain-specific cluster validity measures to find the best result from the set of clustering results. We have conducted experimental evaluation using both synthetic and real datasets and our results show that the weighted coverage density approach powered by the SCALE framework can efficiently generate high quality clustering results in a fully automated manner.

Categories and Subject Descriptors

I.5.3 [**Computing Methodologies**]: Pattern Recognition-Clustering

CIKM'06, November 5–11, 2006, Arlington, Virginia, USA.

General Terms

Algorithms

Keywords

Weighted Coverage Density, AMI, LISR, SCALE

1. INTRODUCTION

Transactional data is a kind of special categorical data, which can be transformed to the traditional row by column table with Boolean values. Typical examples of transactional data are market basket data, web usage data, customer profiles, patient symptoms records, and image features. Transactional data are generated by many applications from areas, such as retail industry, e-commerce, healthcare, CRM, and so forth. The volume of transactional data is usually large. Therefore, there are great demands for fast and yet high-quality algorithms for clustering large scale transactional datasets.

A transactional dataset consists of N *transactions*, each of which contains varying number of *items*. For example, $t_1 = \{milk, bread, beer\}$ and $t_2 = \{milk, bread\}$ are three-item transaction and two-item transaction respectively. A transactional dataset can be transformed to a traditional categorical dataset (a row-by-column Boolean table) by treating each item as an attribute and each transaction as a row. Although generic categorical clustering algorithms can be applied to the transformed Boolean dataset, the two key features of such transformed dataset: large volume and high dimensionality, make the existing algorithms inefficient to process the transformed data. For instance, a market basket dataset may contain millions of transactions and thousands of items, while each transaction usually contains about tens of items. The transformation to Boolean data increases the dimensionality from tens to thousands, which poses significant challenge to most existing categorical clustering algorithms in terms of efficiency and clustering quality.

Recently, a number of algorithms have been developed for clustering transactional data by utilizing specific features of transactional data, such as LargeItem [21], CLOPE [23], and CCCD [22]. However, all of the existing proposals suffer from one obvious drawback. All proposed clustering algorithms require users to manually tune at least one or two parameters of the clustering algorithms in order to deter-

mine the number of clusters to be used by each run of the algorithm, and to find the best clustering result. For example, LargeItem [21] needs to set the support θ and the weight w, CLOPE [23] has a repulsion parameter r, and CCCD [22] has a parameter $MMCD$ as threshold on clusters merging. Unfortunately, the settings of all these parameters are manually executed and are different from dataset to dataset, making the tuning of these parameters extremely hard. No existing proposals, to the best of our knowledge, have offered general guideline for adequately setting and tuning these parameters.

Also there lacks of cluster validation methods to evaluate the quality of clustering, because clustering is an unsupervised procedure. Some generic measures or the interactive visualization method [7] have been developed for clustering numerical data based on statistical and geometrical properties [14]. Due to the lack of meaningful pair-wise distance function, entropy-based measure has been widely used as a generic measure for categorical clustering [5, 18, 8]. However, such general metrics may not be effective as far as specific types of datasets are concerned, such as transactional data. It is recognized that meaningful domain-specific quality measures are more interesting [17, 14]. Surprisingly, very few of the existing transactional data clustering algorithms have developed transaction mining specific clustering validation measure.

In this paper we present a fast, memory-saving, and scalable clustering algorithm that can efficiently handle large transactional datasets without resorting to manual parameter settings. Our approach is based on two unique design ideas. First, we introduce the concept of Weighted Coverage Density (WCD) as intuitive categorical similarity measure for efficient clustering of transactional datasets. The motivation of using weighted coverage density as our domain-specific clustering criterion is based on the observation that association rules mining over transactional data is inherently related to density-based data clustering [15]. Thus we define the weighted coverage density based on the concept of frequent itemsets [3].

Second, we develop two transactional data specific evaluation measures based on the concepts of large items [21] and coverage density respectively. Large Item Size Ratio (LISR) uses the percentage of large items in the clustering result to evaluate the clustering quality. Average pair-clusters Merging Index (AMI), applies coverage density to indicate the structural difference between clusters.

We implement the weighted coverage density clustering algorithm and the two clustering validity metrics using a fully automated transactional clustering framework, called SCALE (Sampling, Clustering structure Assessment, cLustering and domain-specific Evaluation). The SCALE framework is designed to perform the transactional data clustering in four steps, and it can handle transactional datasets that are small or medium or large in size. In the first step it uses sampling to handle large transactional dataset, and then performs clustering structure assessment step to generate the candidate "best Ks" based on sample datasets. The clustering step uses the WCD algorithm to perform the initial cluster assignment and the iterative clustering refinement, until the WCD of the clustering result is approximately maximized. A small number of candidate clustering results are generated at the end of the clustering step. In the domain-specific evaluation step, we apply the two domain-specific measures (AMI and LISR) to evaluate the clustering quality of the candidate results produced and select the best one. We have conducted experimental evaluation using both synthetic and real datasets. Our results show that the weighted coverage density approach powered by the SCALE framework can generate high quality clustering results in an efficient and fully automated manner.

The rest of the paper is organized as follows. Section 2 gives an overview of some related work. Section 3 details the definitions of key concepts used in our clustering algorithm, the algorithm description and complexity analysis. The two measures AMI and LISR for clustering results evaluation are presented in Section 4. We briefly introduce the SCALE framework in Section 5 and report our initial experimental evaluation results in Section 6. We summarize our contributions in Section 7.

2. RELATED WORK

A number of algorithms have been developed for categorical data clustering in recent years [4, 8, 5, 12, 11, 13, 16, 18]. Some algorithms have studied distance-like pair-wise similarity measures, such as K-Modes [16] and ROCK [13]. While it is commonly recognized that a pair-wise similarity (e.g., cosine measure, the Dice and Jaccard coefficient, etc.) is not intuitive for categorical data, there have been algorithms using similarity measures for a set of records. The typical set-based similarity measures are based on information theory, such as expected-entropy in Coolcat [5], MC [18] and ACE [8], mutual information based similarity in LIMBO [4] and information bottleneck in [20], and minimum description length in Cross Association [6]. These algorithms have been focused on generic clustering structure of categorical data. Another observation is that most of these categorical clustering algorithms determine the number of clusters k explicitly. For example, from the earliest k-modes [16] and ROCK [13] to the latest COOLCAT [5], LIMBO [4] and MC [18], k is an input parameter of the algorithm. Assuming the k at the beginning is extremely difficult in practice.

There are also some works on using bipartite graph theory to cluster transactional data [1, 19, 10, 24, 9]. Clustering algorithms based on partitioning bipartite graph usually generate co-clustering results, where columns and rows of the dataset are partitioned at the same time. If they are applied to transactional data, items and transactions are clustered simultaneously, which unnaturally splits the clusters that overlap over a few frequent items. Furthermore, the graph-based algorithms are often memory and time consuming, thus inappropriate for clustering large transactional datasets.

3. WCD CLUSTERING ALGORITHM

In this section, we present the WCD clustering algorithm for transactional data. The key design idea of the WCD algorithm is the definition of the "Weighted Coverage Density" based clustering criterion, which tries to preserve as many frequent items as possible within clusters and controls the items overlapping between clusters implicitly.

This section is organized as follows: First, we introduce the concept of Coverage Density (CD) and Weighted Coverage Density (WCD) as intra-cluster similarity measures. The coverage density measure approximates the naive uniform item distribution in the clusters and is primarily used

to describe the difference between item distributions, while the weighted coverage density measure describes the frequent-itemset preferred item distribution in the clusters and is used in clustering to preserve more frequent itemsets. We also compare CD and WCD based on their connection to statistical and information theoretic methods. Finally we define the WCD based cluster criterion, present an overview of the WCD clustering algorithm, and provide complexity analysis of the algorithm.

3.1 Notations

We first define the notations of transactional dataset and transactional clustering result used in this paper. A transactional dataset D of size N is defined as follows. Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of items, D be a set of N transactions, where transaction t_j $(1 \leq j \leq N)$ is a set of items $t_j = \{I_{j1}, I_{j2}, \ldots, I_{jl}\}$, such that $t_j \subseteq I$. Let $|t_j|$ be the length of the transaction. A transaction clustering result C^K is a partition of D, denoted by $C_1, C_2, \ldots, C_K$, where $C_1 \bigcup \cdots \bigcup C_K = D, C_i \neq \phi, C_i \bigcap C_j = \phi$.

3.2 Intra-cluster Similarity Measures

In this section we illustrate the concept of Coverage Density (CD) and the concept of Weighted Coverage Density (WCD) as intra-cluster similarity measures. To provide an intuitive illustration of our development of these concepts, let us map the transactions of D onto a 2D grid graph. Let the horizontal axis stand for items and the vertical axis stand for the transaction IDs, and each filled cell (i, j) represents the item i is in the transaction j. For example, a simple transactional dataset {abcd, bcd, ac, de, def} can be visualized in Figure 1.

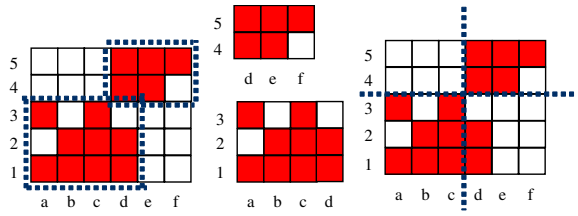


Figure 1: An example 2D grid graph

If we look at the filled area in the graph carefully, two naturally formed clusters appear, which are {abcd, bcd, ac} and {de, def} indicated by two rectangles in Figure 1. In the original graph there are 16 cells unfilled, but only 4 in the two partitioned subgraphs. The less the unfilled cells are left, the more compact the clusters are. Therefore, we consider that the problem of clustering transactional datasets can be transformed to the problem of how to obtain the minimized unfilled number of cells with appropriate number of partitions. Here, if we try to use bipartite graph based co-clustering method to partition the transactions and the items, the result is shown by two straight lines in the right most part of Figure 1. Obviously co-clustering will result in the association loss between item c and item d. This is one of the reasons why we define our clustering problem as row clustering not co-clustering in our application context. This simple example also shows that it is intuitive to visualize the clustering structure of the transactions when they have already been ordered in the specific way as shown in the left most of Figure 1. Thus how to order and partition the transactional dataset properly is one of the key issues of our algorithm.

Bearing this intuition in mind, we define the first concept, *Coverage Density (CD)*, for evaluating the compactness of the partitions in terms of the unfilled cells only. In short, *CD* is the percentage of filled cells to the whole rectangle area which is decided by the number of distinct items and number of transactions in a cluster.

Given a cluster C_k, it is easy and straightforward to compute its coverage density. Suppose the number of distinct items is M_k, the items set of C_k is $I_k = \{I_{k1}, I_{k2}, \ldots, I_{kM_k}\}$, the number of transactions in the cluster is N_k, and the sum occurrences of all items in cluster C_k is S_k, then the Coverage Density of cluster C_k is

$$CD(C_k) = \frac{S_k}{N_k \times M_k} = \frac{\sum_{j=1}^{M_k} occur(I_{kj})}{N_k \times M_k}. \quad (1)$$

The coverage density reflects the compactness of a cluster intuitively. Generally speaking, the larger the coverage density is, the higher the intra-cluster similarity among the transactions within a cluster.

However, the *CD* metric is insufficient to measure the density of frequent itemset, since in the *CD* metric each item has equal importance in a cluster. Namely, if the *cell (i, j)'s contribution* to the coverage density consists of *transactional contribution* T_i and the *item contribution* W_j. In *CD*, both transactional and item contributions are uniform, i.e., $T_i = T = \frac{1}{N_k}$ and $W_j = W = \frac{1}{M_k}$. *CD* can be represented as $T_i \times \sum_{j=1}^{M_k} occur(I_{kj}) \times W_j = T \times W \times \sum_{j=1}^{M_k} occur(I_{kj})$, treating each cell with the same importance as shown in Figure 3(a).

Another problem with the *CD* metric is the situation where two clusters may have the same *CD* but different filled-cell distributions. Consider the two clusters in Figure 2: is there any difference between the two clusters that have the same *CD* but different filled-cell distributions? This leads us to develop a heuristic rule for identifying and selecting a better distribution: we consider the cluster with the coverage density focused on the high-frequency items to be better in terms of compactness than the cluster with the same CD but the filled-cell distribution is somewhat *scattered* with respect to all items.

We now introduce the concept of *Weighted Coverage Density (WCD)* to serve for this purpose.

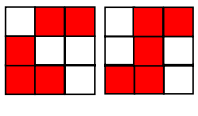


Figure 2: Two clusters with the same CD

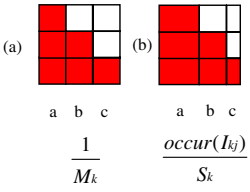


Figure 3: illustration of items contribution

Concretely, we put more "weight" on the items that have higher occurrence frequency in the cluster. This definition implies that the weight of each item is not a fixed one during the clustering procedure and it is decided by the current items distribution of a cluster. Thus, the item contribution W_j is no longer uniform as shown in Figure 3(b). Now, the item contribution W_j is defined as the ratio of occurrences

of each item to the sum of occurrences of all items, namely,

$$W_j = \frac{occur(I_{kj})}{S_k} \qquad st. \sum_{j=1}^{M_k} W_j = 1 \tag{2}$$

By Equation (2), and without changing the transactional contribution T, the *Weighted Coverage Density* of a cluster C_k can be defined as follows:

$$\begin{aligned} WCD(C_k) &= T \times \sum_{j=1}^{M_k} occur(I_{kj}) \times W_j \\ &= \frac{1}{N_k} \times \sum_{j=1}^{M_k} occur(I_{kj}) \times \frac{occur(I_{kj})}{S_k} \\ &= \frac{\sum_{j=1}^{M_k} occur(I_{kj})^2}{S_k \times N_k} \end{aligned} \tag{3}$$

Recall the example in Figure 2, by Equation (3), the weighted coverage density of the cluster on the left is $\frac{9}{15}$, while the weighted coverage density of the cluster on the right is $\frac{11}{15}$. Therefore, the cluster on the right is better, which is consistent with our heuristic rule.

3.3 Comparing CD and WCD

In the above section we have given the intuitive definition of CD and WCD. We will show that CD and WCD are inherently connected to some important statistical concepts.

Let random variable X represent the frequency of an item, we can consider that the occurrences of items in a cluster C_k as the sample probability density function (PDF) of X, denoted by $f(X)$. Therefore, CD and WCD are strongly related to the first moment (the expectation, $E[X]$) and the second moment ($E[X^2]$), i.e.,

$$CD(C_k) = T \times E[X]$$
$$WCD(C_k) = T \times E[X^2]$$

Since $E[X^2] = E^2[X] + Var(X)$, for two clusters that have the same number of transactions (T) and $E[X]$, our clustering criterion of maximizing WCD will prefer the cluster having higher $Var(X)$, i.e., deviating more from the scenario that each item has similar frequency. Let $p(X = I_{kj})$ be the proportion of item occurrences I_{kj} in all item occurrences in cluster C_k. $\sum_{j=1}^{M_k} p(X = I_{kj}) \log p(X = I_{kj})$ is the entropy of this cluster. For M_k number of items, this entropy is maximized when each item frequency is the same, i.e., $Var(X) = 0$, when the variance is minimized. Therefore, maximizing $Var(X)$ is remotely related to minimizing the entropy criterion [5, 18, 8]. We will show that WCD based criterion is more efficient in clustering transactional data, while also generating high-quality clustering results preserving more frequent itemsets.

3.4 Weighted Coverage Density based Clustering Criterion

We define the WCD-based clustering criterion in this section and outline the design of the WCD clustering algorithm in the next section.

To define the WCD-based clustering criterion, we also take into account of the number of transactions in each cluster. For a clustering result $C^K = C_1, C_2, \ldots, C_K$ where $K < N$, we define the following *Expected Weighted Coverage Density (EWCD)* as our clustering criterion function.

$$\begin{aligned} EWCD(C^K) &= \sum_{k=1}^{K} \frac{N_k}{N} \times WCD(C_k) \\ &= \frac{1}{N} \sum_{k=1}^{K} \frac{\sum_{j=1}^{M_k} occur(I_{kj})^2}{S_k} \end{aligned} \tag{4}$$

An EWCD-based clustering algorithm tries to maximize the *EWCD* criterion.

However, if *EWCD* is used as the only metric in clustering procedure, an exception occurs when the number of clusters is not restricted - when every individual transaction is considered as a cluster, it will get the maximum EWCD over all clustering results. Therefore, the number of clusters needs to be either explicitly given or implicitly determined by other parameters. To avoid blindly setting k or tuning complicated parameters, we propose the SCALE framework in Section 5, where a set of candidate "best Ks" is suggested by the BKPlot method [8].

3.5 WCD Clustering Algorithm

The WCD clustering algorithm uses a partition-based clustering approach. It scans the dataset iteratively to optimally assign each transaction to a cluster in terms of maximizing the EWCD criterion. The entire procedure of the WCD-based clustering can be divided into three phases: the clustering structure assessment phase, the WCD based initial cluster assignment, and the WCD-based iterative clustering refinement phase. We call the first phase the WCD algorithm preparation step, which can be performed by using an existing algorithm that can find the best K or best candidate Ks. We refer to the second and third phases as the WCD clustering step, which is executed by the WCD algorithm.

Concretely, in the clustering structure assessment phase, we analyze the given transactional dataset to determine the candidates of critical clustering structure and generate the best K or a few candidate best Ks. One of the possible approaches to perform the first task is to use the Best K method (BKPlot) we have developed [8] for finding the best candidate Ks for clustering categorical datasets. The main idea of the BKPlot method is to examine the entropy difference between the optimal clustering structures with varying K and reports the Ks where the clustering structure changes dramatically. In [8] we developed a hierarchical algorithm that is capable of generating high-quality approximate BKPlots, which can capture the best Ks with small errors. The algorithm also generates a hierarchical clustering tree, where the cluster seeds can be found at different Ks. One can also use other available algorithms that can find the best K in this phase.

In the initial cluster assignment phase, we take the outputs from the clustering structure assessment phase and produce an initial assignment using the WCD algorithm. Concretely, the WCD clustering algorithm takes the K number of clusters and the cluster seeds at the best Ks as inputs to define the initial K clusters. Each seed represents an initial cluster consisting of a few transactions. Given one of the best Ks, the WCD algorithm performs the clustering over the entire dataset. It reads the remaining transactions sequentially, and assigns each transaction to one of the K clusters, which maximizes the EWCD of the current clustering result. Our experiments show that the BKPlot method

can efficiently help reduce the search space and get high quality clustering result.

Since the initial assignment produced in the second phase may not be optimal, in the iterative clustering refinement phase, the cluster assignment is refined in an iterative manner until no more improvement can be made with respect to WCD on the clustering result. Concretely, the algorithm reads each transaction in a randomly perturbed order, and check if the original cluster assignment is optimal in the sense that the EWCD metric is maximized. If it is not optimal, the transaction is moved to currently best fitted cluster, which increases the amount of EWCD the most. Any generated empty cluster is removed after a move. The iterative phase is stopped if no transaction is moved from one cluster to another in one pass for all transactions. Otherwise, a new pass begins. Note that the number of iterations may vary with respect to different random processing sequence and different clustering structure. It also depends on the number of clusters. Our experiments in Section 6 show that two or three iterations are enough for most well structured small datasets, while more iterations are often required for large and noisy datasets.

A sketch of the pseudo code for the WCD algorithm is given in Algorithm 1.

Algorithm 1 WCD.main()

```
Input: Dataset file D of transactions; Number of clusters
K; Initial K seeds
Output: K clusters
/*Phase 1 - Initialization*/
K seeds form the initial K clusters;
while not end of dataset file D do
  read one transaction t from D;
  add t into C_i that maximizes EWCD;
  write < t, i > back to D;
end while
/*Phase 2 - Iteration*/
while moveMark = true do
  moveMark = false;
  randomly generate the access sequence R;
  while has not checked all transactions do
    read < t, i >;
    if moving t to cluster C_j increases EWCD and i ≠ j
    then
      moveMark=true;
      write < t, j > back to D;
    end if
  end while
end while
```

Finding the destination cluster for each transaction is the key step in both phases, which needs to compute/update EWCD for each possible assignment. To avoid unnecessary access and computation, the WCD clustering algorithm keeps the summary information of each cluster in main memory and updates it after each assignment. The summary information of cluster C_k includes the number of transactions N_k, the number of distinct items M_k, the sum occurrences of items S_k, the sum square occurrences of items $S_k^2 = \sum_{j=1}^{M_k} occur(I_{kj})^2$, the distinct items set I_k in cluster C_k, and the occurrences of each item $I_k[j].occur$.

With the summary information, we are able to incrementally compute EWCD. Concretely, the two functions DeltaAdd and DeltaRemove can perform the incremental computing by adding one transaction into a cluster or removing one transaction from it respectively. Since the two functions are similar, we only provide outline of the function DeltaAdd in Algorithm 2. Let $t.I$ be the set of items in the transaction t.

Algorithm 2 WCD.deltaAdd(C_k, t)

```
float deltaAdd (C_k, t)
{
  S_k_new = S_k + |t|;
  ΔS_k^2 = 0;
  for (i = 0; i < |t|; i++) {
      if t.I[i] not exist in I_k then
        ΔS_k^2 ++;
    else
      ΔS_k^2 += (I_k[j].ocurr + 1)^2 − (I_k[j].ocurr)^2;
  }
  return ((S_k^2 + ΔS_k^2)/S_k_new) − (S_k^2/S_k);
}
```

3.6 Complexity Analysis

The space consumption of WCD algorithm is quite small, since only the summary information of clusters is kept in memory. Let K stand for the number of clusters, and M stand for the maximum number of distinct items in a cluster. A total $O(K \times M)$ space is necessary for the algorithm. For a typical transactional dataset with up to ten thousand distinct items, several megabytes will be sufficient for the WCD clustering algorithm.

The most time-consuming part is the update of EWCD to find the best cluster assignment which involves DeltaAdd and DeltaRemove. Since each DeltaAdd/DeltaRemove costs $O(|t|)$, the time complexity of the whole algorithm is $O(\lambda \times N \times K \times |t|)$, where λ is the number of iterations and N is the number of transactions in dataset. Usually λ, K and the length of transaction $|t|$ are much smaller than N, i.e. the running time is almost linear to the size of datasets. So the WCD clustering algorithm is ideal for clustering very large transactional datasets.

4. CLUSTER VALIDITY EVALUATION

We have shown the criterion function *EWCD*, which is approximately optimized by the WCD algorithm with the help of the BKPlot method. In this section, we propose two quality measures for cluster evaluation of transactional data.

LISR− measuring the preservation of frequent itemsets

Since one of the popular applications of transactional data clustering is to find localized association rules [2], we propose a new measure called Large Item Size Ratio (*LISR*) to evaluate the percentage of Large Items [21] preserved by clustering. The more large items are preserved, the higher possibility the frequent itemsets are preserved in the clusters. An item is a "Large Item" when its occurrences in a cluster are above the user-specified proportion of transactions. We name the user-specified proportion as the minimum support, which is remotely connected with the "minimum support" of association rules mining. The *LISR* computing formula is $LISR = \sum_{k=1}^{K} \frac{N_k}{N} \times \frac{LS_k}{S_k}$, where LS_k stands for the total occurrences of Large Items in cluster C_k and S_k stands for the total occurrences of all items in cluster C_k. In the above formula, the number of transactions in each cluster is taken into account in order to reduce the influence of noise tiny clusters to the whole clustering result. A large *LISR* means high concurrences of items and implies the high possibility

of finding more Frequent Itemsets [3] at the user-specified minimum support. In practice, users can provide different minimum supports they are interested for finding association rules, and then compare the *LISRs* of different clustering results to decide which clustering result is the most interesting one.

AMI− measuring inter-dissimilarity of clusters

As we have shown previously, WCD measure evaluates the homogeneity of the cluster and tries to preserve more frequent itemsets, while CD only evaluate the homogeneity. Below we define the heuristic structural difference based on the CD measure, which is shown effective in describing the overall inter-cluster dissimilarity in experiments.

Given a pair of clusters C_i and C_j, the inter-cluster dissimilarity between the C_i and C_j is:

$$d(C_i, C_j) = \frac{N_i}{N_i + N_j} CD(C_i) + \frac{N_j}{N_i + N_j} CD(C_j) - CD(C_i \bigcup C_j) \quad (5)$$

Simplifying the above formula, we get $d(C_i, C_j) = \frac{1}{N_i+N_j}$ $(\frac{S_i}{M_i} + \frac{S_j}{M_j} - \frac{S_i+S_j}{M_{ij}}) = \frac{1}{N_i+N_j}(S_i(\frac{1}{M_i} - \frac{1}{M_{ij}}) + S_j(\frac{1}{M_j} - \frac{1}{M_{ij}}))$, where M_{ij} is the number of distinct items after merging two clusters and thus $M_{ij} \geq \max\{M_i, M_j\}$.

Because of $\frac{1}{M_{ij}} \leq \frac{1}{M_i}$ and $\frac{1}{M_{ij}} \leq \frac{1}{M_j}$, $d(C_i, C_j)$ is a real number between 0 and 1. Here, $S_i(\frac{1}{M_i} - \frac{1}{M_{ij}})$ describes the structural change caused by the cluster C_i. Not surprisingly, when two clusters have the same set of items, that is $M_i = M_j = M_{ij}$, $d(C_i, C_j)$ is zero. For two very different clusters having little overlapping between the sets of items, merging them will result in a large $d(C_i, C_j)$. Therefore, we say the above measure evaluates the structural difference between clusters.

We propose the *Average pair-clusters Merging Index (AMI)* to evaluate the overall inter-dissimilarity of a clustering result having K clusters.

$$AMI = \frac{1}{K}\sum_{i=1}^{K} D_i,$$
$$D_i = Min\{d(C_i, C_j), i, j = 1, \ldots, K, i \neq j\} \quad (6)$$

AMI is the average dissimilarity between all clusters. The larger the AMI is, the better the clustering quality.

Some traditional clustering methods try to optimize the clustering validity measure by combining intra-cluster similarity and inter-cluster dissimilarity [17, 14, 21]. However, this is extremely difficult in practice since we need some domain-specific weighting parameters to combine the intra-cluster similarity and the inter-cluster dissimilarity, and the setting of such parameters may differ from dataset to dataset. Thus, in our prototype implementation of the SCALE framework (see next section for detail), we choose to optimize them separately: we optimize the intra-cluster EWCD measure with the WCD algorithm at the candidate best Ks, and use the AMI measure to select the best one. Our experiments show that AMI is an effective measure for indicating the globally distinctive clustering structure. In addition, LISR is used as a domain-specific measure in the SCALE framework. Our experiments in Section 6 show that the WCD algorithm can generate high quality results reflecting the domain-specific structure.

5. IMPLEMENTING WCD ALGORITHM WITH SCALE FRAMEWORK

We implement the WCD clustering algorithm and the two transactional data specific clustering validity metrics using a fully automated transactional clustering framework, called SCALE (Sampling, Clustering structure Assessment, cLustering and domain-specific Evaluation). The SCALE framework is designed to perform the transactional data clustering in four steps as shown in Figure 4. SCALE uses the sampling step to handle large transactional dataset. Standard sampling techniques are used in the sampling step to generate some sample datasets from the entire large dataset. The framework assumes the primary clustering structure (with small number of large clusters) is approximately preserved with appropriate sample size.

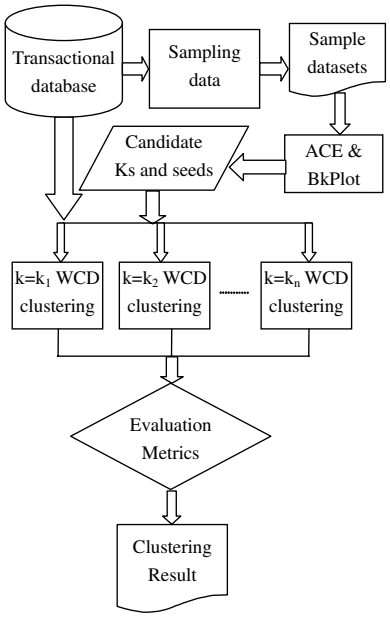


Figure 4: The SCALE framework

In the clustering structure assessment step, SCALE determines the candidates of critical clustering structure and generates the candidate "best Ks" based on sample datasets. In our prototype implementation, the candidates of critical clustering structure are recommended by the Best K method BKPlot developed at Georgia Tech [8]. BKPlot method studies the entropy difference between the optimal clustering structures with varying K and reports only those Ks where the clustering structure changes dramatically as the candidate best Ks, which greatly reduces the search space of finding the domain-specific candidate best Ks. In the SCALE prototype, we use a hierarchical algorithm proposed in [8] to generate high-quality approximate BKPlots, which can capture the candidate best Ks with small errors. The algorithm also generates a hierarchical clustering tree, where the cluster seeds can be found at different Ks. The clustering structure assessment step outputs the best Ks and the cluster seeds at the best Ks to the clustering step.

The clustering step applies the WCD clustering algorithm to perform initial cluster assignment. The initial assessment result is then used to guide the WCD clustering over the entire dataset in an iterative manner until no transaction is

moved from one cluster to another in one pass with respect to maximizing WCD. At the end of iterative assignment refinement, a small number of candidate clustering results are generated. Now we use the domain-specific measures (AMI and LISR) to evaluate the clustering quality of the candidate results produced in the clustering step and select the best one.

We have conducted experimental evaluation using both synthetic and real datasets. The details are reported in the next section. Our results show that the weighted coverage density approach powered by the SCALE framework can generate high quality clustering results in an efficient and fully automated manner.

6. EXPERIMENTS

In this section we evaluate the WCD-based clustering approach using both synthetic and real datasets. Our experiments show three interesting results: (1) the WCD-based measures are effective in determining the meaningful structures of transactional datasets; (2) the WCD algorithm is fast and also capable of generating high-quality clustering results in terms of the domain-specific measures, compared to the existing best-performance transactional data clustering algorithm, CLOPE [23]; (3) the WCD algorithm is scalable to large transactional data in terms of the related factors.

Here, we would like to give a brief introduction to CLOPE [23] at first. CLOPE also maps the item/transaction relationship to a 2D grid graph. However, it defines the intra-similarity of a cluster based on the ratio of the average item occurrences to the number of distinct items. A larger such ratio means the higher intra-cluster similarity. CLOPE computes the profit of a clustering result as $profit_r(C) = \frac{\sum_{i=1}^{K} \frac{S(C_i)}{W(C_i)^r} \times |C_i|}{\sum_{i=1}^{K} |C_i|}$, where $S(C_i)$ is the sum occurrences of items in cluster C_i, $W(C_i)$ is the number of distinct items of cluster C_i and $|C_i|$ is the number of transactions in cluster C_i. The optimal result is achieved when the profit is maximized. The repulsion parameter r is introduced in the profit computing, implicitly controlling the number of clusters. However, there is no guideline to find the appropriate setting of r and we also find this setting may be different from dataset to dataset, which make it difficult to apply CLOPE in practice.

Before reporting the results of our experiments, we first introduce the datesets used in the experiments.

6.1 Datasets

Our experiments have used two synthetic datasets: *Tc30a6r1000_2L* generated by us and *TxI4Dx Series* generated by synthetic data generator used in [3]. In addition, we used two real datasets: Zoo and Mushroom from the UCI machine learning repository [1].

Tc30a6r1000_2L dataset is generated with a two-layer clustering structure that is clearly verifiable, as shown in Figure 5. We use the same method documented in [8] to generate the Tc30a6r1000_2L dataset. We want to use this synthetic dataset to test how well our WCD approach can perform when the critical clustering structures of the dataset are determined correctly at the clustering structure assessment step. It has 1000 records, and 30 columns, each of which has 6 possible attribute values. The top layer has 5 clusters with 200 data points in each cluster, four of which have two overlapping sub-clusters of 100 data points, respectively. In Figure 5, blank areas represent the same attribute value 0, while non-blank areas are filled with randomly generated attribute values ranging from 0 to 5. Since it is a generic categorical dataset, the attribute values are converted to items in order to run the WCD and CLOPE algorithms.

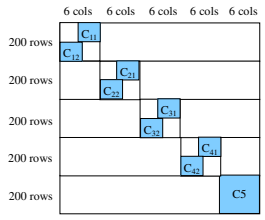


Figure 5: Structure of two-layer synthetic data

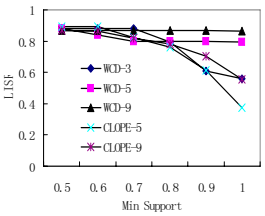


Figure 6: LISR graph for Tc30a6r1000_2L

Zoo Real dataset Zoo from the UCI machine learning repository is used for testing the quality of clustering results. It contains 101 data records for animals. Each data record has 18 attributes (animal name, 15 Boolean attributes, 1 numeric with set of values [0, 2, 4, 5, 6, 8] and animal type values 1 to 7) to describe the features of animals. The animal name and animal type values are ignored in our transformed file, while the animal type also serves as an indication of domain-specific clustering structure.

Mushroom Real dataset Mushroom from the UCI machine learning repository contains 8124 instances, which is also used for quality testing. Each data record has 22 categorical attributes (e.g. cap-shape, cap-color, habitat etc.) and is labeled either "edible" or "poisonous". The dataset contains 23 species of mushroom according to the literature. Therefore, we assume the domain-specific clustering structure could possibly have about 23 clusters. We use these knowledge to assess the clustering results and the effectiveness of the domain-specific measures.

Mushroom100k We also sample the mushroom data with duplicates to generate the mushroom100k of 100,000 instances as a real dataset for performance comparison with CLOPE.

TxI4Dx Series Data generator in [3] is used to generate large synthetic transactional datasets for performance test. We first give the symbols used in order to annotate the datasets. Three primary factors are the average transaction size T, the average size of the maximal potentially large itemsets I and the number of transactions D. For a dataset having $T = 10$, $I = 4$ and $100K$ transactions is denoted as $T10I4D100K$. The number of items and the number of maximal potentially large itemsets are always set to 1000 and 2000. We generate 5 groups of datasets from $T10I4D100K$ to $T10I4D500K$ by varying the number of transactions and each group has 10 randomly generated datasets at same parameters. We also generate 4 groups of datasets from $T5I4D100K$ to $T50I4D500K$ by setting the average length of transactions as 5, 10, 25 and 50. Also each group has 10 randomly generated datasets at same parameters.

[1] http://www.ics.uci.edu/~mlearn/MLRepository.html

6.2 Experimental Results

6.2.1 *Results on Tc30a6r1000_2L*

The candidate Ks generated for Tc30a6r1000_2L at the clustering structure assessment step is {3, 5, 9}. So we run WCD algorithm on $K = 3, 5$, and 9 for Tc30a6r1000_2L. In order to compare WCD algorithm with CLOPE, we run CLOPE with r changing from 1.0 to 3.0 and decide the appropriate r according to the prior knowledge of this synthetic dataset. Since we know the exact clustering structure of Tc30a6r1000_2L, we are able to find the CLOPE clustering results matching the best number of clusters. Without the help of additional knowledge about the clustering structure, it would be impossible to determine r solely by CLOPE documented in [23].

We summarize the quality of clustering results with the measure AMI (Figure 7) and LISR (Figure 6). For CLOPE results on Tc30a6r1000_2L, we show the AMI values for r varying from $r = 1.0$ to $r = 2.5$. The best clustering results include a five-cluster clustering result with $r = 2.0$ and a nine-cluster clustering result with $r = 2.43$. LISR curves show that CLOPE often gives much worse results at higher minimum supports. The cluster-class confusion matrix in Figure 9 also shows that WCD produces better results than CLOPE. The rows of confusion matrix stand for clusters and the columns stand for the original class definition given by the literature.

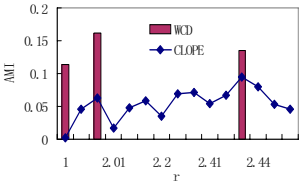


Figure 7: AMI graph for Tc30a6r1000_2L

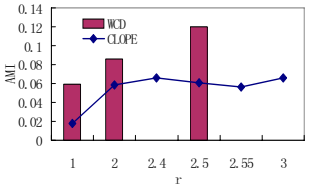


Figure 8: AMI graph for Zoo

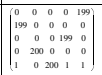
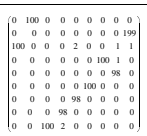
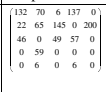
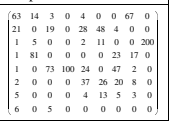
approach	WCD-3	WCD-5	WCD-9
Cluster – Class Confusion Matrix			
		Clope-5	Clope-9
Cluster – Class Confusion Matrix			

Figure 9: Cluster-Class Confusion Matrix for Tc30a6r1000_2L

6.2.2 *Results on Zoo and Mushroom*

The candidate Ks generated at the clustering structure assessment step are {2, 4, 7} for Zoo and {2, 4, 6, 15, 19, 22, 29} for Mushroom. The literature shows the most likely K for Zoo is 7, and it could be around 23 for Mushroom. We run the WCD clustering algorithm on the transformed datasets with different Ks and seeds to get the candidate clustering results. We also run CLOPE on Mushroom with parameter $r = 2.6$ to get 27 clusters, as suggested in [23]. Since there are no reported results of CLOPE on Zoo dataset, we try CLOPE with varying r to find the result that is closest to the domain-specific structure. We find that CLOPE generates 7 clusters only when r is varied from 2.41 to 2.54, and the clustering result at $r = 2.5$ is selected for comparison.

Table 1: AMI for Mushroom

DataSets	approach	K	AMI
Mushroom	WCD	2	0.054954
		4	0.120967
		6	0.116561
		15	0.091988
		19	0.116521
		22	0.081510
		29	0.078027
	CLOPE ($r = 2.6$)	27	0.105191

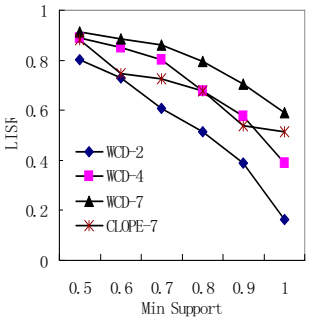


Figure 10: LISR graph for Zoo

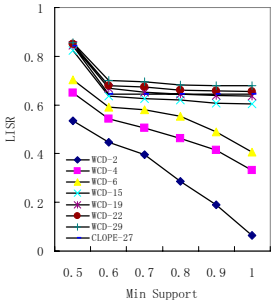


Figure 11: LISR graph for Mushroom

The AMI values in Figure 8 and Table 1 show that the WCD results are better than CLOPE's at the candidate Ks. To compare over different Ks, the AMI index also suggests the WCD result at K=7 is the best for Zoo, and WCD results at K=4, 6, 19 for Mushroom are outstanding.

Figure10 and Figure11 show the comparison of WCD and CLOPE using the LISR measure for Zoo and Mushroom datasets. Consistently, the LISR graphs of Zoo (Figure10) and of Mushroom (Figure11) suggest that the WCD results at K=7 for Zoo and K=19 for Mushroom are the best results. In summary, the experimental results on both Zoo and Mushroom datasets demonstrate that the LISR and AMI measures can help to find the clustering structures that are consistent with the documented structures, and that the WCD approach generates better clustering results than CLOPE, with the additional advantage of no parameter tuning .

6.2.3 *More about* AMI

It is interesting for us to find that *AMI* index is consistent with the results produced in the clustering structure assessment step using BKPlot on some datasets. Figure 7 and 8 have already shown that the *AMI* values at the best Ks suggested by BKPlots for the 2-layer synthetic dataset and the real zoo dataset. Figure 12 and 13 plot the *AMI* curves with varying K for these two datasets, respectively. The

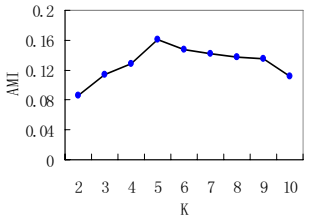


Figure 12: AMI curve for Tc30a6r1000_2L

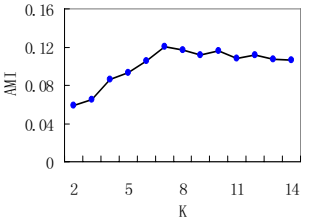


Figure 13: AMI curve for Zoo

global peak values (K=5 for Tc30a6r1000_2L and K=7 for Zoo) always appears at one of the candidate Ks suggested by BKPlot.

6.2.4 *Performance Evaluation on Mushroom100k*

We compare the CLOPE and WCD on the running time of algorithms by varying the number of clusters and by varying the size of dataset.

First, we run CLOPE by varying r from 0.5 to 4.0 with step value 0.5. The running seconds and the number of clusters are reported for each r. The number of clusters is 17, 18, 27, 30, 31, 32, 41 and 64 for different r values, respectively. Correspondingly, we run WCD on these numbers of clusters and get WCD running seconds. The comparison in Figure 14 shows that the cost of WCD is much less than that of CLOPE with respect to the number of clusters produced.

Second, we run CLOPE on 10%, 50% and 100% size of Mushroom100k with $r = 2.0$ and get the number K of clusters are 22, 23, and 30, respectively. Then we run WCD using the same numbers of clusters on the same set of datasets. The results in Figure 15 show that WCD is also much faster than CLOPE with respect to the size of the dataset.

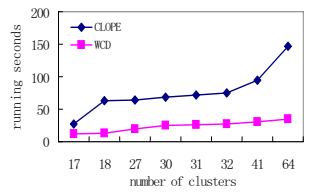


Figure 14: The total running time of CLOPE and WCD on Mushroom100k with varying K

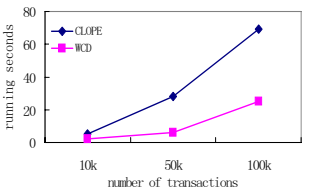


Figure 15: The total running time of CLOPE and WCD on Mushroom100k with varying size

6.2.5 *Performance Evaluation on TxI4Dx Series*

In this section we further study the performance of WCD on large synthetic transactional datasets. The time complexity of WCD algorithm is $O(\lambda \times N \times K \times |t|)$. Since the number of iterations, λ, is not controllable, we study the other three factors: the number of transactions N, the number of clusters K, and the average length of transactions $|t|$. TxI4Dx series are used in this set of experiments.

We first did experiments on 5 groups of T10I4Dx datasets with different size from 100K to 500K and set the number of clusters $K = 10, 50, 100$ separately. Each group has 10 randomly generated datasets with the same parameter setting and we average the running time of 10 datasets as the final running time of each group. Figure 16 and Figure 17 show that the overall cost is almost linear in terms of the size of dataset and the number K of clusters.

Then we did experiments on another 4 groups of TxI4D100K datasets with different average length of transactions and different number of clusters. Figure 18 shows for small number of clusters ($K \leq 50$), the average length of transactions is approximately linear to the running time, while for large K, such as $K = 100$, it becomes nonlinear. Since we are more interested in the clustering structure with small number of clusters ($K \leq 50$), the WCD algorithm is also scalable to the average length of transactions.

7. CONCLUSION

We have presented WCD − a fast, memory-saving and scalable algorithm for clustering transactional data, including two transactional data specific cluster evaluation measures: LISR and AMI. We implemented the WCD clustering algorithm and the LISR and AMI clustering evaluation using SCALE − a fully automated transactional data clustering framework, which eliminate the complicated parameter setting/tuning required by existing algorithms for clustering transactional data. Concretely, the SCALE framework is designed to perform the transactional data clustering in four consecutive steps. It uses sampling to handle large transactional dataset, and then performs clustering structure assessment step to generate the candidate "best Ks" based on sample datasets. The clustering step uses the WCD algorithm to perform the initial cluster assignment and the iterative clustering refinement. A small number of candidate clustering results are generated at the end of the clustering step. In the domain-specific evaluation step, the two domain-specific measures (AMI and LISR) are applied to evaluate the clustering quality of the candidate results produced and select the best one. We have reported our experimental evaluation results with both synthetic and real datasets. We show that compared to existing transactional data clustering methods, the WCD approach (the WCD clustering algorithm powered by the SCALE framework) can generate high quality clustering results in a fully automated manner with much higher efficiency for wider collections of transactional datasets.

8. ACKNOWLEDGMENTS

The first author is partly supported by China Scholarship Council for her one year visit at Georgia Institute of Technology. The second and the third authors are partly supported by grants from NSF CSR, ITR, Cybertrust, a grant from AFOSR, an IBM SUR grant, and an IBM faculty award.

9. ADDITIONAL AUTHORS

Joonsoo Bae (Dept of Industrial & Sys. Eng, Chonbuk National Univ, South Korea, email: `jsbae@chonbuk.ac.kr`) and Zhang Yi(Computational Intelligence Laboratory, University of Electronic Science & Tech. of China, 610054 P.R. China, email: `zhangyi@uestc.edu.cn`).

10. REFERENCES

[1] J. Abello, M. G. C. Resende, and S. Sudarsky. Massive quasi-clique detection. *Latin American Theoretical INformatics*, pages 598–612, 2002.

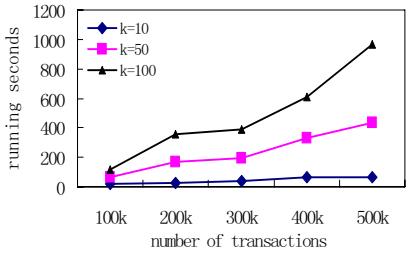


Figure 16: The total running time vs. the size of datasets on T10I4Dx datasets

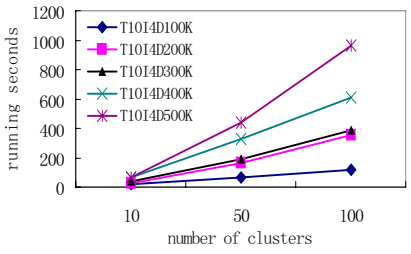


Figure 17: The total running time vs. the number k of clusters on T10I4Dx datasets

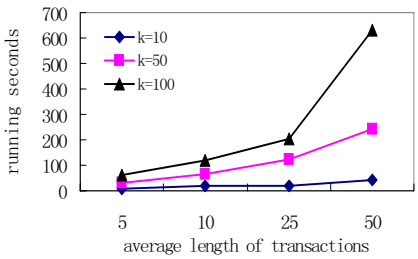


Figure 18: The total running time vs. the average length of transactions.

[2] C. C. Aggarwal, C. Magdalena, and P. S. Yu. Finding localized associations in market basket data. *IEEE Trans. on Knowledge and Data Eng.*, 14(1):51–62, 2002.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499, 12–15 1994.

[4] P. Andritsos, P. Tsaparas, R. J. Miller, and K. C. Sevcik. Limbo: Scalable clustering of categorical data. *Proc. of Intl. Conf. on Extending Database Technology (EDBT)*, pages 123–146, 2004.

[5] D. Barbara, Y. Li, and J. Couto. Coolcat: an entropy-based algorithm for categorical clustering. *Proc. of ACM Conf. on Information and Knowledge Mgt. (CIKM)*, 2002.

[6] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. *Proc. of ACM SIGKDD Conference*, 2004.

[7] K. Chen and L. Liu. VISTA: Validating and refining clusters via visualization. *Information Visualization*, 3(4), 2004.

[8] K. Chen and L. Liu. The "best k" for entropy-based categorical clustering. *Proc. of Intl. Conf. on Scientific and Statistical Database Management (SSDBM)*, 2005.

[9] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, 2001.

[10] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. *Proceedings of ICDM 2001*, pages 107–114, 2001.

[11] V. Ganti, J. Gehrke, and R. Ramakrishnan. Cactus: Clustering categorical data using summaries. *Proceedings of Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 73–83, August 1999.

[12] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: an approach based on dynamical systems. *Proc. of Very Large Databases Conference (VLDB)*, 1998.

[13] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. *Proc. of IEEE Intl. Conf. on Data Eng. (ICDE)*, pages 512–521, March 1999.

[14] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Cluster validity methods: Part I and II. *SIGMOD Record*, 31(2):40–45, 2002.

[15] T. Hastie, R. Tibshirani, and J. Friedmann. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[16] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Workshop on Research Issues on Data Mining and Knowledge Discovery, (DMKD)*, 2(3):283–304, 1998.

[17] A. K. Jain and R. C. Dubes. Data clustering: A review. *ACM Computing Surveys*, 31, 1999.

[18] T. Li, S. Ma, and M. Ogihara. Entropy-based criterion in categorical clustering. *Proc. of Intl. Conf. on Machine Learning (ICML)*, 2004.

[19] N. Mishra, D. Ron, and R. Swaminathan. On finding large conjunctive clusters. *COLT*, pages 448–462, 2003.

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

[21] K. Wang, C. Xu, and B. Liu. Clustering transactions using large items. *Proc. of ACM Conf. on Information and Knowledge Mgt. (CIKM)*, 1999.

[22] H. Yan, L. Zhang, and Y. Zhang. Clustering categorical data using coverage density. *Proc. of Intl. Conf. on Advance Data Mining and Application*, pages 248–255, 2005.

[23] Y. Yang, X. Guan, and J. You. Clope: A fast and effective clustering algorithm for transactional data. *Proc. of ACM SIGKDD Conference*, July 2002.

[24] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon. Bipartite graph partitioning and data clustering. *CIKM*, pages 25–32, 2001.